Representation Methods for an Axiomatic Design Process Software

by

Vigain Harutunian

B.S. Mechanical Engineering
The University of Texas at Austin, 1994

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

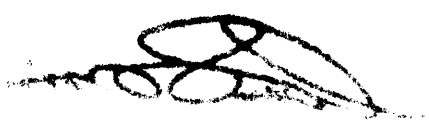
MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 1996

Signature of Author: _____
Department of Mechanical Engineering
August 9, 1996

Certified by: _____
Nam P. Suh
Ralph E & Eloise F Cross Professor &
Mechanical Engineering Department Head
Thesis Supervisor

Accepted by: _____
Ain A. Sonin
Chairman, Departmental Committee on Graduate Studies



Representation Methods for an Axiomatic Design Process Software

by
Vigain Harutunian

Submitted to the Department of Mechanical Engineering
on August 9, 1996 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

Four types of knowledge representation, relational, inferential, inheritable, and procedural knowledge, are investigated within Axiomatic Design (AD). These knowledge types provide the framework to analyze existing representation methods and propose additional methods to be automated by the Axiomatic Design Software (ADS) currently being developed by the AD Group at MIT. Case studies are presented to illustrate the AD method and indicate the need for structured representation methods for design knowledge within industrial design projects.

The ADS is being designed within the axiomatic design process. The functionality and program elements of the software, including the database and graphical user interface, are presented. The software automates the proposed representation schemes to document the development of new design information, based on given design constraints and higher level decisions. Additional research on creating databases for each design domain (customer, functional, physical, and process) is required to develop a thinking design machine (TDM) which can recommend solutions. The structure of each supporting database is dependent on the representation scheme of the AD process. Thus, the representation scheme presented in this thesis can contribute to the development of a TDM.

Thesis Supervisor: Nam P. Suh
Title: Ralph E & Eloise F Cross Professor & Mechanical Engineering Department Head

Acknowledgments

The author acknowledges Professor Nam P. Suh for his vision and guidance throughout this research project. His will to explore the design process and human creativity has motivated the development of the Axiomatic Design Process and the Thinking Design Machine (TDM).

The author acknowledges the Charles Stark Draper Laboratory for funding this thesis work as part of the Fundamental Research in Concurrent Engineering Project.

The author acknowledges Silicon Valley Group, Inc. for allowing the author to implement the Axiomatic Design process on sight and develop a case study as a result of the visit.

The author acknowledges all the members, past and present, of the Axiomatic Design Group at MIT. This group has consisted of several students and professors throughout the 1980's and 1990's whose work has contributed to the current body of knowledge within Axiomatic Design.

The author acknowledges Derrick Tate and Mats Nordlund. The author thanks these fine gentlemen for their assistance , expertise, and contagious curiosity.

The author acknowledges his family for their emotional, physical, and spiritual support through the years of education which lead to this thesis. Their ever-present strength and undoubted love have instilled in the author a wonderful sense of promise and confidence in life.

Dedication

To My Family

The Harutunians

My "Babbé Harut" House Members are:
Ardash, Ossanna, Shafika, Takoohy, Hagob, Kegham,
Shant, and Anne Harutunian.

A special dedication to my Mom, Shafika Harutunian.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Engineering design involves the creative use of physical knowledge to satisfy societal needs ([25], p. 5). Therefore, the field of engineering design finds itself between the physical sciences and the social sciences and thus requires an understanding of both fields (Figure 1.1). According to Suh, "knowledge from design and other fields should eventually converge, forming a continuum of knowledge with few discontinuities, rather than the islands of knowledge that exist today" [24].

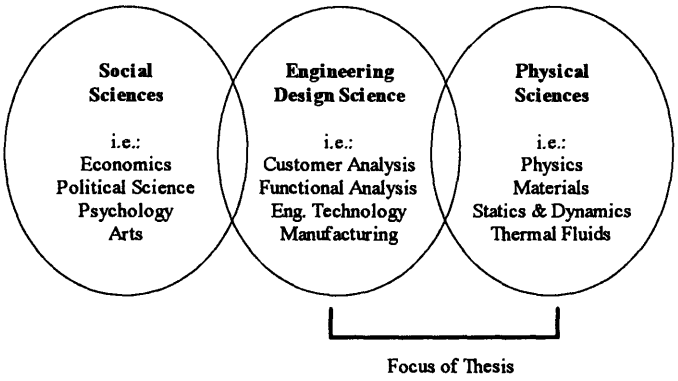


Figure 1-1. Engineering design science with respect to social and physical sciences.[1]

The design method presented in this thesis is the Axiomatic Design method developed by Nam P. Suh at the Massachusetts Institute of Technology [25]. The specific goal of this thesis is to improve the transfer of information between axiomatic design and the physical sciences to promote a continuum of knowledge. To achieve this goal, a representation scheme is required for design science which can be

[1] Dixon and Penny present a similar model in which entering design is placed at the center of *cultural* and *technical* fields [19].

merged with the representation in the physical sciences. This can be achieved by improving the knowledge representation within the design and physical fields and merging of the representation schemes together. The focus of this thesis is on the representation of design knowledge and not on physical knowledge.[2] Thus, the thesis investigates knowledge representation within axiomatic design and proposes additional representation schemes which step toward the merger of representation schemes between design science and the physical sciences.

Please note, the relation between the social sciences and engineering design is outside the scope of this thesis. Section 1.3 presents the author's perspective on the limitations of structured representation methods with respect to the social sciences, such as the arts.

1.1 Motivation for a Structured Design Process

The false hope is that by understanding the physical world as best we can, we can somehow directly use this understanding to develop wonderful products and services to serve human needs. Unfortunately, good knowledge of the physical world does not necessarily yield a good product. The best technology and resources can lead to poor results. Therefore, design processes developed in engineering research are required to apply technical knowledge from our physical world to satisfy customer needs.

The engineering design process includes recognizing and formalizing needs, ideating and creating solutions, and analyzing and testing solutions through an iterative process ([25], p. 27). According to Clausing, "the difference between success and failure can originate in [the products'] responsiveness to customer needs, the viability of the core concepts, the producibility of their design, the robustness of their functional quality, the economical precision of their production, their success of integration, their effective reusability, and their strategic impact" ([4], p. 6). Thus, the success of a product clearly requires more than an understanding of the physical sciences.

There is a need to create a structured design process. According to Suh, one important need to codify the design process is to transfer this knowledge to succeeding generations. Suh states, "If we have to treat every bit of [design] knowledge as an isolated fact, and store the knowledge in its raw, ungeneralized form, the required data base will become so immense that it cannot be retained or managed, even with the most ideal information-processing system" ([25], p.10). Thus, a systematic and concise design process can make it possible to pass large amounts of design knowledge to future generations.

Ulrich and Eppinger propose three needs which a systematic design process can provide for successful product development. A systematic design process provides 1) an explicit decision making method that is well understood by all team members, 2) a checklist to ensure that important steps are not forgotten, and 3) a natural documentation of the decisions made for future reference [29]. Thus, a

[2] For an example of a representation scheme of the physical world, please refer to the bond graph representation scheme [11].

systematic design process can ensure that all steps are taken during the project and help communicate information during and after the completion of a project.

There is also a sound business need for a structured design process. Given today's technologically driven market, the quality of products and services is highly dependent on the technology chosen to meet the customer's needs. Therefore, a sound method is required to identify the true customer needs and find the best set of solutions to address these needs. However, markets are not only becoming more competitive, the number of new technologies is growing every year. As new technologies evolve, there is an increased need to employ sound design principles to assess these technologies based on customer needs. As technologies become commodities, what separates a successful company from the rest is a sound and structured design process.

1.2 Development of an Axiomatic Design Environment

The objective of the author's research is to develop a consistent representation scheme within the Axiomatic Design (AD) process that may help implement the method in industry as part of a design work environment. A major project within this scope of work is the development of an Axiomatic Design software that may automate the AD process and thus help designers gather data, make decisions, and manage the project. In order to transfer the AD theory from the mind of the design engineer to the software environment, the theory must be represented more formally and algorithms for each aspect within the AD process should be developed to create a complete automated design software environment. The representation scheme presented in this thesis is therefore a part of this effort. Chapter 7 describes the software development effort and the implementation of the representation scheme within the software.

The following chapters introduce representation methods which are needed to automate the design process through the software. However, these discussions are limited to the Axiomatic Design model presented in this thesis[3]. This thesis does not cover issues which relate to information flow within the design process. All sources of information are assumed as givens and the thesis only covers how information is represented within the AD framework[4].

1.3 A Perspective on Design

It is appropriate to take a step back and question the notion of design before delving further into the representation of the design process. The design process is a sequence of events which we as humans perform as a response to the world around us. However, the human perception of the world is a very nebulous topic because it is inherently dependent on human understanding, which in-turn, is a dynamic

[3] For a more thorough perspective of the types of algorithms and processes required to automate the generation of design data, please refer to [28].
[4] For more information on the different types of information and information sources required within the design process, please refer to [16].

and evolving body of knowledge. Therefore, it would be difficult to state if we can ever completely model this process of perception. However, we can propose a simplified method of perception.

1.3.1 Discretization

Discretization is a process we commonly perform when trying to understand a large and complex event. The real world is a continuous set of events and boundaries we declare within the physical word are simply a result of our method of perception. In the context of knowledge representation and object oriented modeling, we can identify a set of individual class objects within an event and determine the attributes of each object and how these objects relate to one another [23]. From this representation, we may infer different types of knowledge and eventually learn to control or mimic such events. This method of perception is a logical method of breaking an event into simple components until we reach a set of physical primitives.

However, there are certain events in the world which cannot be easily perceived through discretization. How would a drawing or emotional feeling be perceived through discrete objects? The notion of discretizing a complex event is very personal to our being and, within our society, distinguishes the "arts" from the "sciences". We cannot completely discretize why certain drawings produce fantastic emotional sates, why a verse from Shakespeare sounds pleasant, or why one consumer product seems more appealing than another. Again, the crux of this difficulty is the ever changing perception of the human mind and society. Fashions, trends, mentalities, and cognitive perceptions are ever-changing and are difficult to dicretize. Therefore, it is difficult to represent knowledge within the social realm. This lack of representation makes it a challenge to bridge the gap between the social sciences and engineering design science.

1.3.2 A Graphical Analogy

Imagine if we are to model a phenomenon in nature, a real function that we have observed, as shown in Figure 1.2 (a). This function is real and continuous in a 2-dimensional space. Imagine we do not have a formal model for this function and we are trying to quantify this event somehow. If we are able to measure and quantify the event, we can do so at certain intervals and thus dicretize the entire event until we have a complete map to describe it. Therefore, we can establish a measurement scheme at each axis and develop a formal mapping of the event, as shown in Figure 1.2 (b).

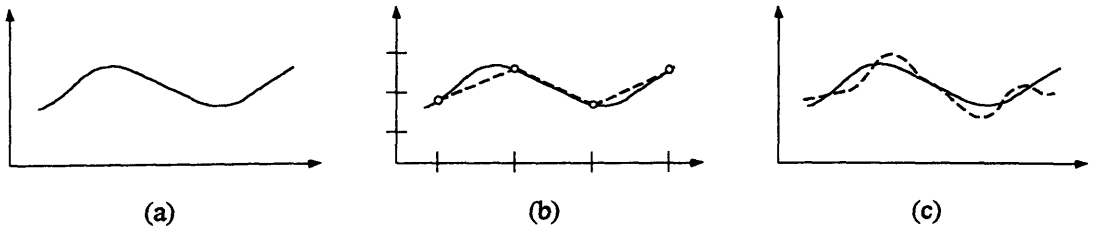


Figure 1-2 Examples of modeling: (a) a real event in the physical world, (b) a discretized model (dashed lines), (c) and a holistic model (dashed lines).

However, discretization assumes axis are defined and measurement with respect to these axis is possible. Assume we cannot easily measure and quantify the event, yet we have some ability to determine whether our model of the event corresponds with the actual event. This is the case when we do not have quantifiable metrics and can only produce qualitative perceptions of the event. How would we attempt to model the event? Given our limitations, we can only use an iterative approach. For the specific example of the function, we can create the function (assuming we can re-create the phenomenon) and determine (again, by some ability, i.e.: qualitative means) if the model we have created matches the real phenomenon we are observing. Therefore, an attempt to mimic the function is to simply create a function which resembles the real event and, afterwards, determine if the new function matches the original event, as shown in figure 1.2 (c). This is termed a holistic approach since the event is modeled as a whole with less insight and information, yielding a more iterative modeling process.

1.3.3 An Artistic Analogy

An artist may have an emotion that he/she wishes to express. The artist is completely aware of the feeling, though the artist may not necessarily quantify this emotion with any of the representation methods we have created (linguistics, mathematics, etc.). However, the artist may choose some medium and attempt to recreate the emotion within this medium. This mapping, which is not understood because the primitives of the source space are not well defined (the artist's mind), will be iterative. The primitives are like the axis in the graphical analogy. It may take more than one attempt to completely express the feelings inside the artist's mind because there is no direct mapping. This is the case of the holistic model in Figure 1.2 (c).

However, once the object is created in a medium which has a set of primitives that may be quantified, such as a physical painting or sculpture, we can quantify and perform operations on this object. We have developed a set of mathematical tools and measurement systems that allow us to model events within a subset of the physical world for which we can define primitives (such as length, time, mass, color, etc.). Thus, we can re-scale the object, change the color, sharpness, and perform many other physical operations on the resulting object, given the primitives of these operations are understood. This case is like the discretized model shown in Figure 1.2 (b) in which the axis are well defined.

1.3.4 Documentation of Holistic Models

As a result, designs within a space of non-quantifiable primitives are harder to document and decompose than designs within a space of quantifiable primitives. If the space is poorly understood, it would be difficult to discretize the space and decompose the design into smaller subcomponents. At best, one could only hope to document each iteration of the effort. However, because there is no established set of primitives, documentation may be inconsistent. Therefore, in the case of holistic models, the criteria for selecting the best model will be inconsistent and thus the documentation of the selection process will be vague.

1.4 Scope of Design Covered

Because the AD process is object oriented and requires an understanding of the elements in the physical design as part of the zigzagging process, the remainder of this thesis will focus on the design which is quantifiable. Therefore, it is not suggested that the AD process can produce a better painter or sculptor. However, the method can be used to design objects with known attributes which may be decomposed into subsystems with inherent functionality. To say that the AD process cannot be used in very "artistic" processes, is over-presumptuous. At best, the author will state that such artistic process, because they are not well quantifiable, will result in AD hierarchies which contain many branches which cannot be further decomposed.

It is possible that the AD method can be used as a general mapping approach between two spaces independent of the representation of these spaces, discretized or continuous. For example, if the artist understands his or her feelings, though cannot quantify them through some representation, they may still follow a general mapping process and principles described within the AD method. Therefore, an artist may continue to decompose within the AD hierarchy. But, with little representation, it is likely that only the original artist will be able to follow the decomposition and thus the hierarchy will only serve as documentation for the original artist. Therefore, the remainder of this thesis covers the application of design to engineering problems in which the information is quantifiable and the primitives for the design space may be established such that the design may be measured with respect with these primitives.

1.4.1 Scope of Design Methods Covered

Over the past four to five decades, researchers in the field of engineering design have sought to develop a better understanding of the design process. As a result, several types of methods and processes have emerged around the world, especially in the United States and Europe. In retrospect, the work has resulted in two types of design models, prescriptive and descriptive. Prescriptive models provide activities and rules which, according to the design model, a design process should follow. The descriptive models

focus on capturing how design is done and the models are formalized by studying how designers work and think [6].

This thesis deals with the class of methods that are prescriptive. The Axiomatic Design method consists of rules in the form of axioms, corollaries, and theorems that are indicative of good designs. In addition, the method focuses on product attributes to assess the designs. It is interesting to note that these rules are a result of Professor Nam P. Suh's experience in the field of design and manufacturing; an experience of observation and learning from actual product attributes. Therefore, these were generated after studying successful *products* and identifying attributes which distinguish good products from bad products [25].

1.4.2 Representation Schemes in Engineering Design

There are numerous approaches to the representation of engineering design data, including methods from artificial intelligence and the engineering disciplines. The bulk of the research is on managing design information and is not specific to the representation of information within one given design method. Therefore, the bulk of the research deals with issues that apply to the storage and retrieval of design information. This is critical research which has lead to the development of numerous algorithms including systems for describing design artifacts [7], methods of constraint negotiation [13], the Data Manipulation Language (DML), integrity constraint checking programs , and SHADES, a program which incorporates a product data manipulation language[8]. In addition, there are products on the market that help organize design information known as PDMs, product data managers. Such systems help associate product components with part numbers, vendors, and other project computer files, such as solid modeling drawings, FEA, and CFD files.

Much of these algorithms and methods are focused on data within the physical domain. However, they do not incorporate the zigzag process[5]. This thesis puts more emphasis on the design process and the representation of information as a mapping between the functional and physical domain.

1.5 Outline of Thesis

The approach of this thesis is to start with a design method, Axiomatic Design, and build a representation scheme based on the fundamentals of this method. The following list summarizes the content of each chapter to follow.

- Chapter 2 provides a background on the AD method. Topics covered are the fundamentals of domains, the zigzag process, and the axioms.

[5] Please refer to Chapter 2, section 2.1.2, for more information on domains and the zigzag process in design.

- Chapter 3 identifies the types of knowledge representation that exist within the AD method. Four types of knowledge representation are presented and illustrated within the AD framework.
- Chapter 4 provides a case study to illustrate the axiomatic design process. A case study on electronic packaging covers the gathering of customer needs, the definition of FR's and the selection of DP's.
- Chapter 5 proposes enhancements to the representation structure within the AD method. The goal of the enhancements is to improve the representation of design information and to facilitate the development of new design information.
- Chapter 6 provides an industrial case study on semiconductor process equipment design. The case study focuses on the development of FR's and DP's based on given design constraints for a semiconductor process equipment machine.
- Chapter 7 presents the development of an AD software. The chapter covers the needs for such a software, the development process which uses the AD method, and the functionality of the software based on representation schemes presented in this thesis.
- Chapter 8 proposes an implementation of the representation methods as part of a TDM.

Chapter 2

Introduction to Axiomatic Design

The axiomatic design process provides a set of axioms to the field of design, a field which some consider more of an art than a science. The hard sciences were all once mystical before the advent of axioms and theorems within their fields. Euclid, Kepler, Newton, Einstein, and other giants of science have transformed our physical understanding of the world to what is now modern science. The aim of axiomatic design is to do the same to the field of design. Instead of building and designing artifacts on an ad hoc basis, the axiomatic design process provides a structured process for making decisions during the design process. The method defines the fundamental knowledge structure for information within the design process and the appropriate flow of information within this knowledge structure. The method also provides a set of axioms to select the appropriate solution based on the given information. As a result, the method provides a means of documenting design rationale for redesign and troubleshooting.

2.1 The Structure of Design Information

The following section presents the information structure within the AD process. The section covers the domains of design information, the elements within each of these domains, the relations among elements within each domain, and the order in which information is entered among the domains.

2.1.1 Domains

Information within the design process may be sorted into domains. A domain of knowledge can be viewed as a certain perspective regarding the design process. There are four domains defined within the AD process: 1) the customer domain, 2) the functional domain, 3) the physical domain, and 4) the process domain. These domains are general and may apply to any design situation, but the types of elements contained within each domain is specific to the type of problem being addressed [26].

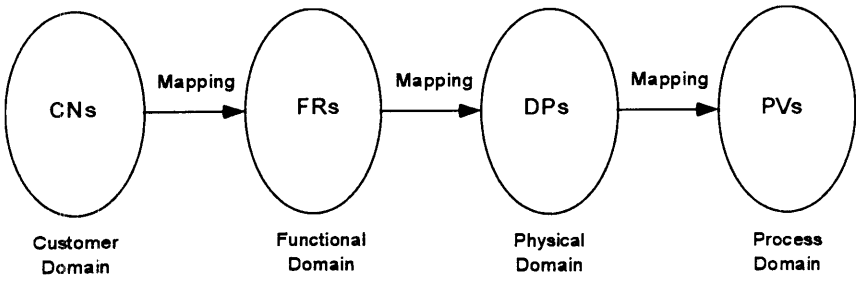


Figure 2-1. Design domains.

In general, information flows from the customer domain to the process domain. Each pair of domains have a *what* and a *how* relationship. The domain on the left contains *what* is to be achieved, and the adjacent domain to the right contains *how* this is to be achieved. Therefore, the domain on the right satisfies the domain on the left. Each *what* and *how* relationship is a mapping process, as shown in Figure 2.1.

The customer domain holds information obtained from the customer known as *Customer Needs (CNs)*. The information the customer provides pertaining to the product is placed within the customer domain. In industry, such information may be collected by the marketing department and placed in marketing documents.

The functional domain contains the *Functional Requirements (FRs)* which depict the functionality of the product as defined by the design engineers. An FR is defined as the minimum set of requirements which completely characterize the design objectives for a specific need.

The physical domain contains the *Design Parameters (DPs)* which describe the intrinsic nature of the design artifact to satisfy the FRs. Information from the physical world may be gathered, for example, by experienced design engineers who can analyze the artifact within the physical context or through standard references, catalogues, or databases.

The process domain contains the *Process Variables (PVs)* which explain how the DPs are produced. For example, if the design artifact were a mechanical product, the information for the process domain would originate from the manufacturing engineers and the machines within the manufacturing line.

2.1.2 Design Hierarchy and Zigzagging

Each domain, with the exception of the customer domain, has a structured *design hierarchy*. The design hierarchy represents the decomposition of elements within each domain, from a system to a sub-system level. The child elements within each hierarchy support the parent elements. If the child elements are not satisfied, the parent elements are subsequently not satisfied.

Elements within one level of the hierarchy cannot be decomposed until they have corresponding elements within the adjacent domain explaining how the elements are to be achieved. For example, a set of FRs cannot be decomposed until each FR has an appropriate DP. Once the DPs are chosen, the FRs may be decomposed into sub-FRs (Figure 2.2). This process of moving between the domains at each level of the hierarchy is called *zigzagging* or the *zigzag process*. The zigzag process ensures that an element is decomposed only if the what-how mapping of the element is complete.

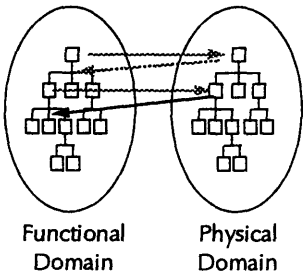


Figure 2-2. Decomposition by zigzagging

The zigzag process is repeated at each level of decomposition within the domain hierarchies. The designer follows the zigzag process, checking the correctness of the design at each level with the design axioms (described in 2.2), until he/she reaches a point in the decomposition in which the solutions to the remaining sub-problems are known. For example, if the parent FR were *to travel*, the mapping process requires the selection of a DP, such as *automobile* or *airplane*, to satisfy this FR. The selection of the DP has a direct impact on the child FRs. For example, the child FRs which result from the DP selection of *automobile* are different from the DP selection of *airplane*.

A naming convention has arisen for the elements within the design hierarchies. Each element is first identified by its type - FR, DP, or PV - and then by a decimal based numbering convention. For example, DP *automobile* would have child elements DP.1 *engine* and DP.2 *frame*. Each of the child DPs can then be decomposed into other elements, for example, engine may be decomposed into DP.1.1 *cylinder* and DP.1.2 *piston*. This numbering convection helps organize large systems of elements within a hierarchy [47]. The assessment of design knowledge within each level of hierarchy is presented in the following section.

2.2 Evaluate Design Knowledge: Design Axioms

This section presents the fundamental axioms used to evaluate a given set FRs, DPs and PVs at each level of the design hierarchy. In this context, design knowledge is the information within the design

domains. Before presenting the axioms, a simple model of the general design process is presented to illustrate the need for such assessment tools.

2.2.1 Design Process

The design process inputs design information, such as constraints and customer needs, and outputs a product which satisfies this information. Figure 2.3 presents a model developed by Wilson which illustrates this process.

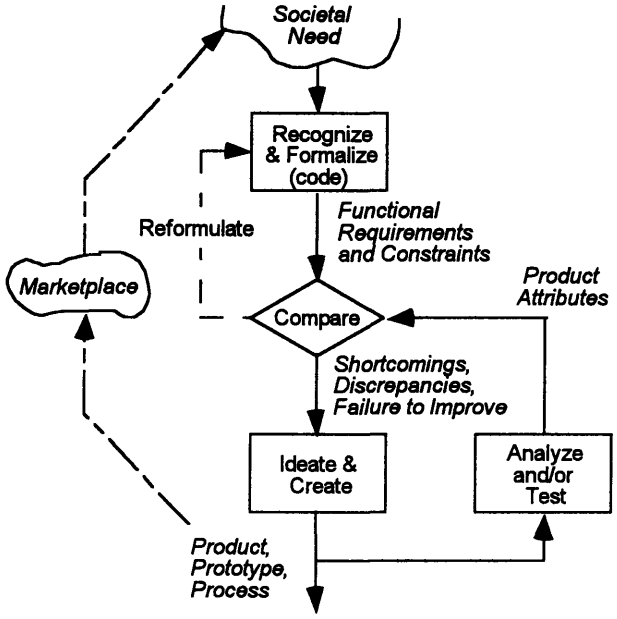


Figure 2-3. The design process according to Wilson [25].

With respect to AD, societal need is contained within the customer domain and the activity labeled "recognize and optimize code" results in FRs and design constraints. However, a major emphasis of this figure is the loop that results after the development of FRs. Once the FRs are established, the designer ideates and creates product DPs, which can then either pass directly to the marketplace or go through analysis and testing. It is this loop of analysis and testing that contains the design axioms.

The design process is therefore iterative. At each level of the design hierarchy, as a new set of DPs are created, the DPs are analyzed to determine if they meet the given set of FRs and design constraints. If they do not, new DPs are developed and the analysis is repeated until the FRs and constraints are satisfied. This model also applies to the DP-PV mapping, in which case, new PVs are developed which must satisfy the given set of DPs. The following section presents the design axioms.

2.2.2 Assessment of Design Elements

The solutions are assessed by the following three conditions in this specified order: 1) the design constraints, 2) the Independence Axiom, and 3) the Information Axiom. Each condition is described in further detail in the following sections.

2.2.2.1 Design Constraints

If the solution does not meet the given set of constraints, it is simply not feasible. This assessment phase is similar to a preliminary feasibility study based on given constraints. Examples of constraints are cost, weight, and time. These constraints should not be confused with FRs, which is a common misunderstanding for students in AD. There are two key attributes that separate constraints from FRs. First, unlike FRs, constraints do not have to maintain independence with respect to one another or other FRs (refer to the first axiom in the following section on the independence of FRs). Second, unlike FRs, constraints do not have tolerances. Constraints are simple bounds on attributes which cannot be exceeded by the solution (i.e.: DPs or PVs).

Design constraints can originate either from outside the design process, known as *input constraints*, or within the design process, known as *system constraints*. Input constraints originate from numerous sources, including customers, the management within design organizations, and the laws of nature and society. System constraints originate from solutions chosen at a higher level within the hierarchy. For example, constraints from parent DPs propagate to child FRs as part of the zigzag process.

In general , a design with a large set of design constraints results in a small set of possible design solutions. Therefore, a negative consequence of a large set of design constraints is limited creativity (the ability to find non-obvious solutions) due the shear reduction of the possible set of solutions. However, on the positive side, a large set of design constraints does consequently reduce the development time by limiting the number of solutions that may be considered. A large number of constraints reduces the number of analysis cycles within the Wilson model (Figure 2.3) and thus reduce the overall development time ([25], pp. 39-40).

2.2.2.2 The Independence Axiom

The first axiom states [25]:

The Independence Axiom (First Axiom):

Maintain the independence of functional requirements.

The first axiom applies to a set, or vector, of FRs and their associated mapped vector of DPs. A *vector* of elements within each level of a hierarchy is defined as a set of elements that have the same parent

element. The first axiom is applied to a pair of vectors from the functional and physical domains in the form of a design matrix A:

$$\begin{Bmatrix} FR_1 \\ : \\ FR_n \end{Bmatrix} = [A] \begin{Bmatrix} DP_1 \\ : \\ DP_n \end{Bmatrix}. \tag{2.1}$$

The elements of the design matrix are determined from the set of equations ([25] p. 122):

$$\begin{aligned} \Delta FR_1 &= \frac{\partial FR_1}{\partial DP_1}\Delta DP_1 + \ldots + \frac{\partial FR_1}{\partial DP_n}\Delta DP_n \\ &\vdots \\ \Delta FR_n &= \frac{\partial FR_n}{\partial DP_1}\Delta DP_1 + \ldots + \frac{\partial FR_n}{\partial DP_n}\Delta DP_n \end{aligned} \tag{2.2}$$

where each element in the design matrix A is defined as:

$$A_{ij} = \frac{\partial FR_i}{\partial DP_j}. \tag{2.3}$$

The above equations also apply to the DP to PV mapping. The variables in equations 2.2 and 2.3 should be changed to reflect the physical to process mapping. FRs are changed to DPs and the current DPs are changed to PVs. The design matrix B is defined as follows:

$$\begin{Bmatrix} DP_1 \\ : \\ DP_n \end{Bmatrix} = [B] \begin{Bmatrix} PV_1 \\ : \\ PV_n \end{Bmatrix}. \tag{2.4}$$

According to the independence axiom, the elements in the *left hand* vector within the design equations should remain independent. Independence is violated if there are off-diagonal terms in the design matrix. Therefore, if a *change* in DP_j has an affect on the ability to achieve FR_i (where $i \neq j$) under the given tolerances, FR_i is not independent within the system of FRs and DPs. Thus, according to equation 3, element A_{ij} would be non-zero and may be represented by the letter *X*. Ideally, the design

matrix should only contain elements on the diagonal (Figure 2.4(a)). This arrangement results in an *uncoupled* design which satisfies the independence axiom.

Elements off the diagonal complicate the mapping process among two vectors. For example, an off-diagonal element within an FR to DP design matrix implies FR_i requires information from DP_j prior to selecting DP_i. As a result, the development of DP_i is dependent on the development of DP_j, and thus DP_j must be developed prior to DP_i. Therefore, if the first axiom is not satisfied, the off-diagonal elements create functional coupling which restricts the development of DPs during initial mapping and subsequent changes to DPs due to a change in FRs. The degree of functional coupling and the affect on the development of DPs is dependent on the arrangement of off-diagonal elements in the design matrix.

A design matrix with elements on only one side of the diagonal results in a *decoupled* design which conditionally meets the independence axiom (Figure 2.4(b)). The condition is that the elements in the left hand vector must be achieved in a specific order, thus the elements on the right hand side must be altered in a certain sequence. For example, a decoupled matrix means the FRs can be satisfied if the DPs are varied in the right sequence. The DPs are positioned in this proper sequence (from top to bottom) if all the elements in the deign matrix are positioned *below* the diagonal.

If there are elements both above and below the diagonal for all possible sequences of elements, then the design is *coupled*. This is an undesirable design because it requires continuous DP iteration to meet the given set of FRs. Therefore, a coupled design is unstable because a change in one DP can possibly have catastrophic affects on the set of FRs and thus require an iterative sequence to change the DPs to once more satisfy the FRs.

$$\begin{Bmatrix} FR_1 \\ FR_2 \end{Bmatrix} = \begin{bmatrix} X & O \\ O & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \end{Bmatrix} \qquad \begin{Bmatrix} FR_1 \\ FR_2 \end{Bmatrix} = \begin{bmatrix} X & O \\ X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \end{Bmatrix} \qquad \begin{Bmatrix} FR_1 \\ FR_2 \end{Bmatrix} = \begin{bmatrix} X & X \\ X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \end{Bmatrix}$$

(a) Uncoupled (b) Decoupled (c) Coupled

Figure 2-4. Examples of the three types of design matrices.

2.2.2.3 *The Information Axiom*

After the design is analyzed through the first axiom, the second axiom is applied. The second axiom is the information axiom and states [25]:

The Information Axiom (Second Axiom):

Minimize the information content [required by the design].

The design information is defined as the information content that is related to the probability of achieving the given FR (in the case of an FR to DP mapping). For an uncoupled set of n FRs, information I may be defined as [25]:

$$I = \sum_{i=1}^{n} [\log \frac{1}{p_i}] \tag{2.5}$$

where p is the probability of success of DP_i achieving FR_i and the log is either a natural log or the log based on 2. The total information is the sum of the information content of each FR. The total information is a simple summation because for an uncoupled design, satisfying each FR is a mutually exclusive event. The information content represents the amount of information *required* (not provided) by a DP to achieve an FR. Therefore, the least amount of information (greatest probability of success) is the best design [24].

The probability of satisfying the FR through a DP (or satisfying a DP with a PV) can be plotted graphically. Figure 2.5 presents a plot of the *design range (dr)* specified by the FR and *system range (sr)* inherent to the DP. The vertical axis represents the probability density function and the horizontal axis is for the FR (or DP for physical to process mapping). The overlap between the system range and the design range is called the *common range (cr)*.

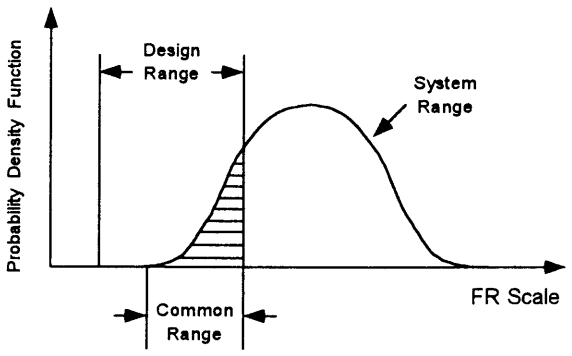


Figure 2-5. Plot of design range, common range, and system range [24].

The ratio between the system range over the common range results in the probability of satisfying the FR (or DP for physical to process mapping). Thus, the information content I for the FR can be calculated from the plot and is equal to:

$$I = \log\left(\frac{A_{sr}}{A_{cr}}\right) \tag{6}$$

where A_{sr} is the area under the system range and A_{cr} is the area within the common range.

Please refer to *The Principles of Design* [25] for more detail on the axiomatic design method. The text contains additional theorems and corollaries which are derived from the two axioms. The text also provides several case studies which illustrate the application of these axioms in fields ranging from the design of mechanical and electrical systems to organizations.

2.3 Conclusion

Axiomatic design provides a framework for describing design objects and the tools by which they may be assessed. The framework consists of several design perspectives, called domains, a structured hierarchy to facilitate information flow within each domain, and a zigzag process to construct the hierarchies in parallel among domains. The axioms provide a consistent set of analysis tools to assess each design decision. Thus, the method provides a consistent documentation scheme to describe design objects and the intent behind the objects. Such a structured framework provides an environment in which new problem formulations and old designs can be closely investigated such that new innovative solutions can be created.

Chapter 3

Knowledge Types within Axiomatic Design

Knowledge representation is a well established subset of artificial intelligence (AI). The author's goal is to apply representation methods to automate mundane tasks which at the moment, given our current level of computer technology, computers can do better. Thus, the goal is to give the designer leverage so that he/she can have the time and means to design larger and more complex systems. Computer networks, databases, and interactive graphical interfaces provide the means for automating many design tasks which have hampered the design process.

This chapter provides a background on the current level of representation in axiomatic design and serves as an introduction to the enhancements presented in Chapter 5. The background includes definitions of basic knowledge types and examples of these types within axiomatic design. The knowledge types presented are used in the development of an axiomatic design software presented in Chapter 7.

3.1 Background on Types of Knowledge Representation

Four types of knowledge representation are 1) relational knowledge, 2) inheritable knowledge, 3) inferential knowledge, and 4) declarative/procedural knowledge [22]. The author does not intend to present these four types of knowledge as the definite set of knowledge types, if such a set were to exist. Instead, the intention of this categorization is to provide a structure to analyze knowledge representation within axiomatic design.

3.1.1 Relational Knowledge

Relational knowledge provides a framework to compare two different objects based on equivalent attributes ([22], p. 109-110). Any instance in which two different objects are compared is a relational type of knowledge. For example, physical relations in engineering relating the force and acceleration of a

mass, or financial relations in banking relating inflation to spending are examples of relational knowledge. Relational knowledge representation is common in database systems and proves to be powerful in sorting different objects relative to their attributes.

3.1.2 Inheritable Knowledge

Inheritable knowledge is, by definition, knowledge obtained from associated objects. This form of knowledge prescribes a structure of objects in which new objects are created which may inherit all or a subset of attributes from existing objects ([22], p. 110-113). For example, the more recent forms of computer programming, such as object oriented programming (OOP), incorporate this notion of inheritance into their programming scheme [23].

3.1.3 Inferential Knowledge

One may infer information from objects through the known relations among objects. Knowledge from relations among a set of objects is called inferential knowledge [22]. Reading a word in context is one familiar example of this type of knowledge. A word alone is simply syntax, but with the help of other objects, such as other words within a phrase, the reader may infer more from a word. This inference within linguistics is called semantics.

3.1.4 Declarative and Procedural Knowledge

A *declarative statement* is one in which knowledge is specified, but the use to which that knowledge is to be put is not given ([22], p. 171). A *procedural representation* is on in which the control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself ([22], p. 172). According to Rich and Knight, there has been a long debate about the difference between procedural knowledge and declarative knowledge ([22], p. 173). Examples of declarative knowledge are laws, the periodic table, and people's names. They are facts which can stand alone and are not dependent on other knowledge. Examples of procedural knowledge include computer programs, directions, and recipes. This type of knowledge indicates a specific type of use or implementation.

It is difficult to determine whether or not the knowledge inherently implies how it is to be used. The difficulty comes in when we analyze each of these examples in finer detail. For example, one could argue that a procedural computer program is a set of declarative statements. Or, that declarative laws constitute a legal process. Although there is a relative difference between declarative and procedural representation, the absolute difference, for all cases, is not so clear. Examples of these two methods in AD are discussed in section 3.2.4.

3.2 Types of Knowledge within AD

The following sections identify examples of knowledge representation within axiomatic design. Each section presents a specific knowledge type and examples within different domains.

3.2.1 Relational Knowledge within AD

Within AD, relational knowledge is primarily used to associate elements of one domain with elements from another domain or set of design constraints. The result of this knowledge type is a mapping of elements among different domains and the entry and use of constraints within the design process. Thus, there are relations among elements within domains due to mapping, and relations among elements within the design domains and the set of design constraints. Three specific examples are considered in this section: 1) relations between elements within the customer domain and functional domain, 2) relations between elements within the functional domain and the physical domain, and 3) relations between elements within the physical domain and the set of design constraints.

3.2.1.1 CN to FR Mapping

The voice of the customer, as recorded in the customer domain, must be related to the functional requirements specified by the designer. As each FR is defined within the functional domain, a relation must be drawn between the new FR and an element, or set of elements, within the customer domain. This relational knowledge is vital to ensure 1) all customer needs are satisfied, and 2) unnecessary FRs are not created.

3.2.1.2 FR to DP Mapping

A second example of relational knowledge is the mapping between the functional domain and physical domain (the mapping between FRs and DPs). In this case, relational knowledge between these elements is used 1) to ensure each FR has a one to one mapping with a DP, and 2) to identify relations among FRs and non-intended DPs. The first form of relational knowledge ensures each FR is satisfied by a DP, and conversely, each DP is justified with an FR. The second form of representation ensures the design is functionally feasible and provides knowledge to populate the design matrix.

3.2.1.3 DPs and Design Constraints

A third example of relational knowledge is the relation between the physical space and the set of design constraints. When discussing design constraints, it is important to remember that design constraints are obtained from the four domains and may influence the development of new elements within the domains. Therefore, within the physical domain, constraints 1) evolve from design parameters, and 2) dictate which DP is most feasible from a set of possible DPs during the first pass of DP selection.

Developing Constraints from DPs. Constraints are related to their source, such as a design parameter, if that were the source. These relations are vital to 1) justify the constraint in case it should be questioned and 2) relate the DPs to their associated constraints in case a DP were to change and thus its associated constraints were to change as well. Maintaining a consistent and accurate set of constraints is critical because the set of constraints influence the development of DPs and the development of elements in the other domains.

Selecting DPs Based on Constraints. In design, relational knowledge can be used to rank DPs based on different criteria when selecting the most feasible DP from a set of possible DPs. For example, different types of constraints, such as cost, weight, or torque, may serve as criteria for the selection process of DPs. Table 3.1 below presents a set of wrenches that may serve as a set of possible DPs and their attributes. If the set of design constraints were to include a maximum cost of $10 and a maximum weight of 1.5 pounds, we would conclude from the table below that only one DP is feasible - Wrench A.

Table 3.1: Simple relational knowledge

DP's	Cost ($)	Weight (lb.)	Torque (lb. ft.)
Wrench A	10	1.4	110
Wrench B	12	0.9	90
Wrench C	9	1.7	130

Design information can flow from one domain to another through the different types of relational knowledge discussed. For example, let us assume the customer wants to travel from point A to point B. Therefore, the main FR is to travel between these points. From this FR, the designer can select a DP from a set of options. However, the customer may also provide constraints, such as time (both duration of flight and time of departure), cost, and comfort level. These constraints are applied during the DP selection process. This example is a simple illustration of the CN to FR and FR to DP mappings and the transfer of constraints along the customer, functional and physical domains.

3.2.2 Inheritable Knowledge within AD

Inheritable knowledge is embodied by the design hierarchies found in the functional, physical and process domains. Within the hierarchy, elements inherit attributes from their parents. The prescription of inheritable knowledge is left to the designer, and, as in many cases of inheritable knowledge, not all attributes of the parent element may be prescribed to the child elements. The types of

attributes passed down to child elements include 1) functional coupling within higher level DMs and 2) design constraints.

3.2.2.1 Functional Coupling

As described in the Chapter 2, functional coupling results when the first axiom is not satisfied and is represented by an off-diagonal element within the design matrix. Coupling not only indicates a poor design at the current level of the hierarchy, it complicates the *development* of new FRs and DPs at the next level of the hierarchy. First, coupling requires the abstraction of more knowledge from the parent DPs when developing new FRs. Second, coupling restricts the sequence in which new FRs are generated and satisfied.

Knowledge required for sub-FRs. As part of the zigzag process, sub FRs are developed with the knowledge of the DP which satisfies the parent FR. Consider the example shown in Figure 3.1 below. According to the zigzag process, elements FR.2.1 and FR.2.2 require knowledge from element DP.2. In addition, because there is coupling (FR2 is affected by DP1 as shown by the off-diagonal element in the deign matrix), elements FR.2.1 and FR.2.2 require knowledge from DP.1 as well. Therefore, the formulations of FR.2.1 and FR.2.2 are influenced by both DP.2 and DP.1. In contrast, FR.1.1 and FR.1.2 only require information from DP.1. The design matrix determines which higher level DPs influence the development of new lower level FRs. Therefore, knowledge from the higher level design matrix is required at each level of decomposition before defining lower level FRs.

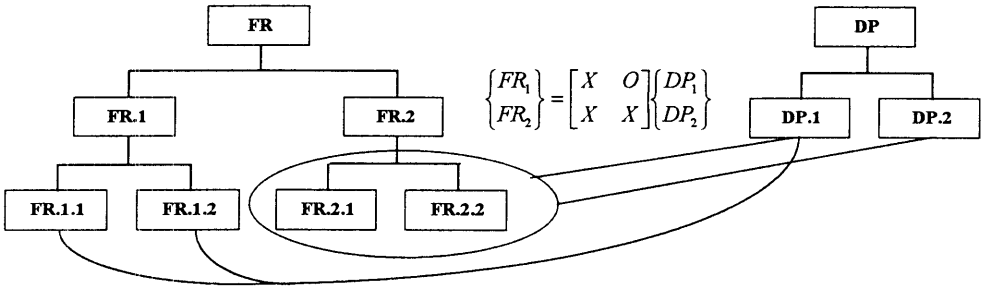


Figure 3-1. The affect of functional coupling on the development of sub-FRs[6].

Development sequence of sub-FRs. The sequence in which lower level FRs are decomposed is also dependent on information from the higher level design matrix. The main requirement is that an FR cannot be decomposed unless it has been satisfied by a DP (the zigzag process). Therefore, the decomposition of FRs is dependent on the development sequence of DPs, which, in turn, is dependent on

[6] Lindhom [14] presents a similar discussion on the development of lower level FRs.

the design matrix. For example, as a result of the coupling in the design matrix shown in Figure 3.1, element DP.2 is developed after element DP.1. Therefore, FR.2 is decomposed only after DP.1 *and* DP.2 are both developed. Note, the design matrix in Figure 3.1 does not place a restriction on the decomposition of FR.1. Element FR.1 is independent of DP.2 and can be decomposed after DP.1 is established.

3.2.2.2 Design Constraints

A second type of inheritable knowledge is design constraints. Constraints can originate from the customer domain, the physical and process domains (as system constraints), and input constraints (i.e.: from corporations, industry standards, law, etc.). All constraints passed to the functional hierarchy trickle down the hierarchy, such that constraints are passed down from parent to child. The constraint propagation process is illustrated in Figure 3.2 below.

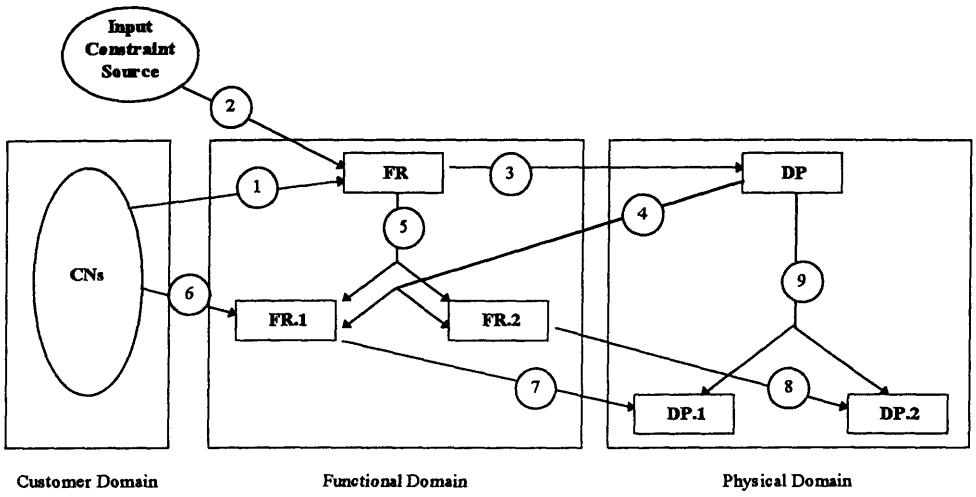


Figure 3-2. The propagation of constraints.

Figure 3.2 illustrates the large number of constraints which propagate within one decomposition. Each solid line has a number inside a circle, m, and represents a unique set of n constraints $(C_1, C_2, \ldots, C_n)_m$. In the figure, set 1 originates from the customer domain and becomes a part of the functional domain. In addition, a set 2 originates from another input constraint source, such as industry standards, and is applied to the top level FR. All the appropriate constraints are then applied to the physical domain during the DP selection process as set 3.

Set 4 originates from the top-level DP and is applied to the next level FRs: FR.1 and FR.2. Constraints applied to the top-level FR (sets 1 and 2) are inherited by the child FRs (set 5). Constraints may be applied from the customer domain to lower level FRs any time during decomposition (set 6). A

set of constraints (sets 7 and 8) are applied to the lower level DPs (DP.1 and DP.2, respectively). Constraints resulting from the top-level DP are passed down to lower level DPs (set 9).

The above example illustrates the large amount of information transfer between two levels within the hierarchy and the need for a structured representation of constraints within the zigzag process. Potential methods are presented in chapter 5. In addition, referring back to Figure 3.1, FR.1 and FR.2 inherit system constraints from element DP as part of the zigzag process. However, elements FR.2.1 and FR.2.2 inherit system constraints from both DP.1 and DP.2 because of the coupling that exists within the higher level design matrix. Therefore, as discussed in section 3.2.2.1, the design matrix also influences the development of new FRs. Note, the design matrix is not shown in Figure 3.2 to simplify the illustration.

3.2.3 Inferential Knowledge within AD

Inferential knowledge is powerful in that new information may be generated from given information. Therefore, this new information does not require further data gathering from the source, but does require *analysis* of the given information to generate new knowledge. The following sections illustrate inferential knowledge within 1) the functional domain and 2) the set of design constraints. The section also includes a discussion on inference methods and their general application to axiomatic design.

3.2.3.1 Functional Domain

An FR can serve as an example of inferential knowledge within axiomatic design. If an FR is not directly provided, the designer must infer this knowledge from the other domains and set of constraints. New FRs may evolve as inferential knowledge from one source, such as the physical domain, or, in more complex situations, from several domains and constraints at once. Chapter 4 presents a case in which FRs are derived from given constraints. In general, the development of FRs is not trivial and is the key to a successful design. Inference for FRs is complicated and requires a look at all of the given information from all domains and set of design constraints. The analysis of this information ranges from numerical computation to human judgment.

3.2.3.2 Constraints

Constraints may be inferred from any of the design domains. For example, when prescribing constraints to an FR, the constraints may be inferred from the customer domain and physical domain. Consider the FR *transport*. Constraints associated with *transport* may be inferred from the customer needs. For example, if the customer wishes 1) to travel to a certain *destination*, 2) to *view the country side* during transport, and 3) arrive within a limited *time*, such information may be inferred as constraints. Constraints may also be inferred from DPs. For example, if the higher level DP were to attend a certain

event, all the attributes of this *event*, such as the *time* and *location*, would be inferred as system constraints and may apply to lower level FRs. The inference may not be direct. For example, depending on the type of *event* or customer need, the designer may infer additional constraints regarding the *style* of transport. Thus, the inference of constraints may be more complex and require more than one source domain.

3.2.3.3 Inference Methods and Representation

Inferential knowledge can take on different forms. From mathematical to logical, different methods of inference have been developed to obtain knowledge from a set of objects. Given a set of mathematical relations and values, one may infer other values or relations, as is the case with Gaussian elimination. If not enough knowns are given, the inference leads to a new set of relations instead of pure values. In any case, the method does yield some new knowledge from the givens.

In addition to algebraic relations, the use of *predicate logic* (mathematical deduction) may be used to infer information from a given set of attributes. This form of reasoning is similar to mathematical proofs in that new statements are validated through older more established statements as the basis for new claims. Inference through predicate logic uses a set of logical operations to relate individual data. Namely these logic symbols are: "$\rightarrow$" (material implication), "$\neg$" (not), "$\vee$" (or), "$\wedge$" (and), "$\forall$" (for all), and "$\exists$" (there exists). S. H. Kim and Nam P. Suh developed a computer algorithm written in PROLOG (a computer language which automates predicate logic operations) to develop a more mathematical representation of the design axioms[7].

The following are examples of predicate logic statements which we will take as givens within some context.

1. Wonder is the name of a dog: **Wonder(dog)**
2. All dogs belong to the class of animals: $\forall$**x: dog(x)** $\rightarrow$ **animal(x)**
3. All animals either live on land or in water.$\forall$**x: animal(x)** $\rightarrow$ **live (x, Land)** $\vee$ **live (x, Water)**

Therefore, we can infer from these three statements that Wonder lives either on land or in water. As more information is made available about these objects and their relations, more knowledge can be inferred[8]. The remainder of this thesis will focus on the general implications of knowledge representations for modeling DPs, constraints, and FRs as part of the design process and will not discuss the specific implementation of predicate logic in AD.

---

[7] Please refer to [25] (Chapter 10) for a complete description of this work.
[8] Please refer to [2] for a more extensive explanation of predicate logic.

3.2.3.4 Implementation of Inferential Knowledge Methods in Engineering Design

As a concluding note on the power of inference, inferential knowledge depends on the quality and availability of attributes and relations among given objects. A computer aided design environment is capable of inferential knowledge only if there exists a thorough database and a consistent documentation system. Therefore, the inference method is useless if there is no substantial amount of design information.

In addition, the inference method is highly dependent on the granularity of information. For example, predicate logic can model information within some limited scope of granularity and allow a designer to infer new information within that scope. However, the use of predicate logic at the conceptual design phase of a complex and abstract design process can be cumbersome and impractical given the lack of detailed design information. Therefore, the effectiveness of an inference method is highly dependent on the granularity of information it requires and thus an inference method may not always scale-up well to a level which provides more abstract information.

A famous linguist, Noam Chomsky, makes a valid point with respect to the ability to infer knowledge between two different scales. The following passage is from Chomsky's work entitled *Language and Thought*. In the passage, he discusses the unification of knowledge between molecular chemistry and the cognitive sciences:

> To the extent that unification [of different fields of knowledge] has been achieved, it has not been achieved through strict reductionism, except in rare cases. It's very commonly been the case that what we think of as more the fundamental science had to be radically revised.... It's perfectly true that the gap between molecules and mind is vast, but the gap between molecules and just about everything else in biology is also vast. It's striking to see how when you get above big molecules, understanding tails off very quickly, and you get into hand waving, or else just description. The problem is that, beyond the level of big molecules, you just don't understand very much; there isn't much in the way of theory [3].

The discussion also covers the unification of theories, but we will limit the scope to the levels of knowledge Chomsky refers to. The relation between the actions of molecules and their implications at a higher level on human thought are not well understood. Therefore, though it may be theoretically possible to infer how the mind works from molecular chemistry, we currently cannot scale-up the inference rules of chemistry to the cognitive sciences.

The analogy also holds in the case of abstract and detailed levels of design. The analogy refers to the abstract level of design (analogous to cognitive sciences) where predicate knowledge is not practical and the detailed level of design (analogous to molecular chemistry) where the use of predicate logic is practical. In design, many high level decisions, such as the selection of top level FRs and customer needs, require high level inference which cannot be achieved by a detailed set of predicate logic rules. Thus,

within the conceptual design phase of a complex design project in which there is a lack of information or the lack of a common representation scheme, inference methods that are broadly structured, such as people skills, human judgment, and empirical data, may prove to be more effective than a detailed set of predicate logic rules which cannot scale-up to consider all facets of the project.

3.2.4 Declarative and Procedural Knowledge within AD

As discussed earlier in section 3.1.4, the difference between declarative and procedural knowledge is not always clear. However, there are definite cases of each type within AD. Examples of declarative knowledge are the design axioms and domains. The axioms are taken as a given, and by definition, are assumed to be true unless a counterexample is found to invalidate them. Customers, the percieved functionality of products, the physical world, and the means of producing products are represented by the domains and are all knowledge which simply exists within our physical and social context. Therefore, both of these examples - the axioms and design domains - serve as declarative statements that can stand alone.

Examples of procedural knowledge within AD are the mapping process between domains, the zigzag process used to decompose the hierarchies, and the sequence in which the axioms are used. These procedures inherently give a designer the steps which are necessary for design. These procedures use the axioms as tools and the design domains as given information within the design process. Thus, both declarative and procedural knowledge are required for a complete design method.

3.3 Scope of Knowledge Representation Covered in Thesis

The scope of knowledge representation within this thesis is to identify areas within the AD process which require additional representation. Thus, the goal is not to introduce new design processes or methods. Rather, the goal is to improve the representation of design elements by strengthening the relational knowledge amongst domains and constraints, strengthening the inheritable knowledge within the zigzag process, and strengthening the ability to infer information from design elements to consequently generate new design elements. The remainder of this thesis will further investigate these issues and propose representation methods to improve the above listed forms of knowledge representation within axiomatic design.

Chapter 4 presents a case study in which the axiomatic design method is used. The case study focuses on the electronic packaging of computer systems and illustrates the significance of knowledge representation within the design process.

Chapter 4

Case Study: Computer Hardware Design

The following case study illustrates constraints in the framework of Axiomatic Design. The study covers the representation of DPs, the creation of FRs, and the assessment of DPs.

Computer hardware design involves the mounting of the electronic cards, power supplies, fans, and other components that are part of the computer system. For a given project, the design tasks of a mechanical hardware design (also known an mechanical packaging) team are further downstream than most other teams within a computer company. Therefore, a primary task for the team is to gather data from the other teams within the company. Because their development process is highly dependent on the constraints imposed by other teams, it is important to document and track the design constraints throughout the development process. Typical components that induce system constraints include processor cards, memory cards, power supplies, fans, heat sinks, and input/output devices. Common input/output devices include internal cards for peripherals, floppy drives, and cable ports [21].

4.1 Abstraction of Constraints from the Customer Domain

The first task for the design engineer is to gather data from the Customer Domain. Table 4.1 below presents a list of customer needs. It is important when collecting data from customers, that needs are listed in a solution neutral form. This list should not include possible solutions proposed by the customer. Customer suggestion should be recorded and investigated during the DP selection process.

Table 4.1. Customer Needs

	Customer Needs List for a Workstation Computer
1	Easy to move around
2	Occupies little space
3	Quiet
4	Looks Nice
5	Easy to set up
6	Input and output devices (such as floppy drives, buttons, cable ports) are easily accessible

Some of these customer needs may be better quantified as metrics or constraints imposed by the customer. For example, we could determine the decibels of noise the user can tolerate, the area of work space the user may give-up for the computer, etc. Such quantitative data may provides insight to the designer for developing FRs. Table 4.2 lists quantified constraints obtained from the customer needs.

Table 4.2. Customer Based Constraints

	Constraints (Metrics)	Unit
1	Weight	Pounds
2	Size of computer	Cubic Inches
3	Noise dissipated	Decibels
4	Aesthetically appealing	Subjective
5	Set-up time	Minutes
6	Location of input/output devices	Inches

The designer uses these quantitative constraints to determine the feasibility of design parameters. In contrast, the more qualitative constraints, such as aesthetics, require a more iterative design process with the help of design aides such as prototypes and customer feedback interviews. As a note, the contrast between these two approaches of design is based on the quantification of information that is possible and stems from the discretized versus holistic modeling approaches discussed in the introductory chapter, section 1.3.

4.2 Development of Top Level FR.

The top level FR is obtained directly from the customer needs. According to the needs listed in Table 1, the customer seeks a package that will be comfortable. That is, the package must be easy to handle, have peripherals that are easily accessible, and not disturb the living environment (aesthetically, spatially, and acoustically). Therefore, the top level FR is:

FR: Provide a comfortable computer package.

4.3 Design Constraints

The designer considers the given design constraints when searching for DPs to satisfy the FR. Design constraints may be imposed by numerous organizations, including the customers (as presented in Table 4.1), management, fellow engineers, concurrent teams working on the same project, industrial standards organizations, and other companies. Table 4.3 lists some non-customer based constraints.

Table 4.3. Constraints

	Constraint	Unit	Source
1	Tolerance to earthquake vibrations	Test	Industrial Standards
2	Acoustic dissipation based on office environment	Decibels	Industrial Standards
3	Component geometry	Inches	Electrical Department
4	Component power requirement	Watts	Electrical Department
5	Component thermal dissipation	Watts	Electrical Department
6	Component critical operating temperature	Celsius	Electrical Department
7	Component connections	Pounds	Electrical Department
8	Maximum size	Inches	Distribution network
9	Maximum weight	Pounds	Distribution network

Constraints 8 and 9 are also listed in Table 4.2 which contains constraints obtained from the customer needs. Therefore, more quantitative information is required to determine which source, the customer (Table 4.2) or other sources (Table 4.3), provides the greater constraint.

4.4 Development and Selection of Top-Level DPs

The development of DPs is not a trivial matter. In essence, everything done to this point in the design process has dealt with the problem definition, without regard to any particular solution DP.

4.4.1 Descriptive vs. Prescriptive DPs

Design parameters may be classified as *descriptive* or *prescriptive*. If a design parameter exists, the designer may describe all the attributes and these attributes become system constraints for lower level elements. This is a descriptive DP. If a design parameter does not exist, it will be defined, or prescribed, by the designer. This is a prescriptive DP. The information gathered for prescriptive DPs originates from both the FR that is to be satisfied and the given set of constraints. Designs with more prescriptive DPs at the higher level of the hierarchy are more novel and creative while designs with more descriptive DPs are less creative and provide more constraints for the lower levels of the hierarchy. Also, descriptive DPs tend to have attributes that are not desired or that are unnecessary and exist as "extra luggage". Because the designer cannot create *everything*, the lower level elements in the DP hierarchy are inevitably descriptive.

An example of a *descriptive* DP for the mechanical package at this stage within the design process is a prepared computer box which is designed to handle various cards and components. An example of a *prescriptive* DP is a new custom box that may be developed to satisfy all of the given design constraints. Each alternative DP is denoted by a letter:

DP_a: Prepared computer box

DP_b: New custom computer box

4.4.2 The Selection of DP's

The next step is to decide which of the two DPs is acceptable under the given set of constraints. A table of design constraints (criteria) is generated to better document the selection process.

Table 4.4. Evaluation of DPs

Constraint	Required Value	DP_a	DP_b
Weight	x lb.	y lb.	z lb.
Size	x cubic in.	y cubic in.	z cubic in.
Acoustic abatement	x decibels	y decibels	z decibels
Time to market	x months	y months	z months
Cost	x $	y $	z $

For the purpose of this discussion, we will assume DP_b is chosen from the two DPs based on the constraints listed in Table 4.4. After selecting the DP, the designer decomposes the top-level FR and defines lower-level FRs. The first axiom is not invoked at the top-level since only one FR-DP pair is defined at that level.

4.5 Development of Lower Level FRs

The creation of lower-level FRs is *dependent* on the top-level DP selected. Therefore, all the attributes of the previous DP should be documented before abstracting the next level of functional requirements.

The development of FRs is of theoretical interest within the field of Axiomatic Design. How are FRs created? The author approaches one avenue within this inquiry. It is proposed that the creation of FRs is based on the ability of the current design to meet the set of deign constraints. Thus, if the current design does not satisfy all of the design constraints, a new FR is created to remedy this conflict. Identifying conflicts among two constraints requires an understanding of the relations among the constraints. Thus, if the values of two constraints do not satisfy the given relation among constraints, then we may say the constraints are in conflict with one another. Constraints are obtained from several sources, including customers, sources external to the design team (input constraints), and higher level DPs (system constraints). Thus, the attributes of a DP become system constraints at a lower level. Therefore, a conflict may arise among 1) attributes of different DPs, 2) attributes of one DP and system constraints, or 3) among a set of given constraints. All three types of conflicts are possible for both descriptive and prescriptive DPs. For example, it is possible that attributes of different DPs within the same DP vector are in conflict and thus warrant a new FR at the next level of decomposition.

A set of conflicting constraints does not necessarily indicate a poor set of constraints or a poor quality of data gathering. There are many instances in which constraints simply do not comply under the given circumstances or relations.

4.5.1 Example of FR Development

For example, if a person's job required that the person be at location A, but the person were at location B, there would be a conflict within the given set of constraints where the source of the first constraint is the person's job and the source of the second constraint is the person's physical location. Here, the relation among these constraints is simple: A and B must be equal. There are several approaches to eliminate this conflict. For example, the person may try to alter the constraint imposed by the job and decide to stay at point B. Or, the person may decide to alter the physical constraint and move to point A. An additional option is to move to point C (where C does not equal A nor B) as some type of compromise, for which both original constraints are affected. Which decision the person makes is dependent on the rigidity of the constraints. In conclusion, understanding the relation between two conflicting constraints is the first step in resolving the conflict. Deciding on what action to take, is synonymous with defining the functional requirement. For example, if the person chooses to alter his or her physical location, the person's FR would be to move from one point to another.

4.5.2 Identifying Lower-level FRs

The development of FRs from a given set of constraints also applies to the mechanical packaging case study. At the outset of design, the electrical teams provide components to the mechanical engineers, along with the set of component attributes. Attributes include thermal dissipation and clamping force requirements and are provided in the form of internal documents. After analyzing the given set of constraints, two subsystems with conflicting constraints are identified: 1) the thermal dissipation of the chips, and 2) the clamping force of the card connectors. The following sections present the details of this analysis.

Thermal dissipation of the chips. The electrical component team provides the chips, complete with substrates and integrated circuit boards. Two system constraints that are the thermal dissipation of the chips (in Watts) and the maximum operating temperature of the chips. These constraints originate from different sources, the electrical and materials departments, and are passed down to the mechanical team to address.

Table 4.5. Development of Thermal Dissipation FR.

System	Chip on IC card
Input Constraints	1. Power generated in chip (from electrical department)
	2. Critical temperature of chip (from materials department)
	3. Temperature of ambient air (industrial standard)
	4. Natural convection coefficient (presumed heat transfer mode)
Relation among constraints	Heat transfer from chip to air (conduction and convection models)
Analysis	Constraints conflict - chip temperature exceeds critical temperature

It is important to understand how power dissipation is related to the temperature of the component. Therefore, the designer generates a model to describe the relationship between the given constraints, such as the simple circuit model shown in Figure 4.1. The designer may use the model to predict 1) how much the critical temperature should rise, 2) how much the thermal dissipation should be lowered, or 3) how much the heat transfer coefficient should be increased.

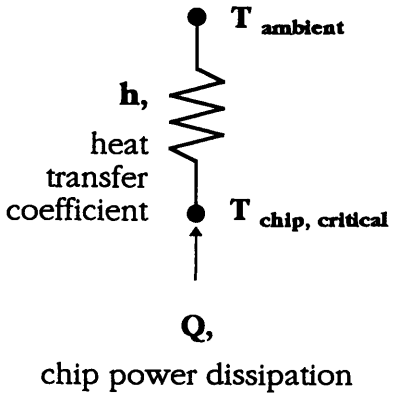


Figure 4-1. Circuit diagram model of thermal dissipation from chip.

In this example, the first three design constraints have rigid sources. Therefore, the designer is left with one option: increase the heat transfer coefficient. This modification on the given design constraints becomes a lower-level FR.

FR1: Increase heat transfer coefficient.

Clamping force of card connectors. The second system involves the clamping force required to maintain electrical contact between a mother board and peripheral cards. The mother board, peripheral cards, and clamping mechanisms are given to the mechanical hardware design department. The task is to determine the additional amount of support required to maintain a working contact between the mother board and peripheral cards. Additional support might be required to meet standard earthquake vibration tests. Therefore, the given constraints are the strength of the clamping unit, the vibrations induced by the test, and the mass of the cards.

Table 4.6. Development of Clamping FR.

System	IC cards and clamping unit
Input Constraints	1. Clamping unit applied force (from clamping department)
	2. Earthquake test induced vibrations (from industrial standards)
	3. Mass of components
Relation among constraints	Clamping force exceeds required force to sustain vibrations
Analysis	Empirical data - clamping force not sufficient to sustain vibrations

The relation for the constraints requires that the connectors have enough clamping force to withstand earthquake vibrations. The purpose is to determine the appropriate FR and find a possible DP. To eliminate the conflict, the designer may choose to increase the clamping strength of the connectors, reduce the mass of the components, or reduce the vibrations transferred to the components. The first option is not possible because of serviceability issues. Though the connectors should provide enough clamping strength to ensure solid connections, they should also be compliant to allow easy docking and removal of components. The second option, to reduce the mass of the components, is also not feasible because most of the components have been designed by the electrical department. Therefore, the connectors and components cannot be modified and the designer has to reduce the vibrations transmitted to the components. This becomes the resulting functional requirement.

FR2: Reduce vibrations transmitted to the components.

4.6 Development and Selection of Lower-Level DPs

Once the FRs are defined, the design engineers locate possible design parameters to satisfy the FRs and design constraints. The following sections step through this development process for the functional requirements identified in the previous section. This process illustrates the importance of constraints for both DP development and selection.

4.6.1 Development of DPs

4.6.1.1 DP1 for FR1: Increase Heat Transfer Coefficient

Once the FR is identified, the first step in developing a DP is to gather all the design constraint information. At this point in the design process, it is possible that new constraints have arisen from various organizations, including management and other design teams, such as the component design teams. For example, once management learns of the new FR1: *Increase Heat Transfer Coefficient*, an additional design constraint may be specified:

Constraint: No liquid coolant shall be used.
Source: Management

Management may justify this constraint through previous experience regarding the reliability and cost of liquid coolant systems. In addition, one of the electrical teams responsible for a thermally critical electrical component may change one of the thermal constraints regarding power dissipation. Therefore, the original set of constraints must be refreshed as new FRs are developed.

After all constraints are reviewed, the designer begins to search for possible DPs to increase the heat transfer coefficient. The different modes of heat transfer (conduction, convection, and radiation) are investigated as part of a feasibility study. Since management has eliminated liquid as a medium for heat exchange, the designer resorts to convection by ambient air as the dominant form of heat transfer.

Assuming the thermal density is very high and natural convection is not feasible (does not satisfy the given constraints), the design engineer investigates different methods of forced convection. The effect of different air flow rates across the surface are modeled to determine the feasibility of forced convection. If this form of heat transfer is not feasible, heat sinks are introduced to increase the surface area ratio between the component's surface area and surrounding air. Heat sinks of different sizes and fin spacing are modeled to determine the feasibility of such a cooling medium.

As a result of this analysis, the designer has considered four different DPs:

DP.1_a: Natural convection (bare component)
DP.1_b: Natural convection with heat sink
DP.1_c: Forced convection (bare component)
DP.1_d: Forced convection with heat sink

4.6.1.2 FR2: Reduce vibrations transmitted to the components

The first step in the DP selection process is to gather all the given constraints. The mechanical packaging engineer consults with the component engineers and determines all of the stresses and strains

the components may withstand. The thermal engineer also communicates with the mechanical packaging engineer to inform him or her of recent developments regarding the weight and spatial geometry of the heat sinks which require support. The communication between the thermal engineer and the mechanical packaging engineer indicates a coupling which is discussed in the following section.

Given these constraints, the packaging engineer may begin generating possible DPs for the FR. To reduce shock transmission, the engineer may choose a common sheet metal frame to dampen any loads or impact exerted onto the components. However, with enough creativity, the engineers may develop different methods to satisfy the FR. Other materials, such as elastic plastics or composites could be used instead of metal. Also, a combination of materials, as found in viscous-elastic structures (layers of viscous-elastic materials sandwiched between sheets of metal) may be used to reduce shock transmitted to the components. Therefore, the designer considers the following set of DPs:

$DP.2_a$: Sheet metal frame

$DP.2_b$: Plastic frame

$DP.2_c$: Composite frame

$DP.2_d$: Viscous-elastic frame

4.6.2 Selection of DPs

At this stage of design, the selection of DPs is a three step process. First, the feasibility of each DP is assessed based on the given design constraints. Second, the design matrix is populated and the first axiom is applied to determine the level of functional coupling within the feasible set of DPs. If more than one set of DPs is acceptable based on the first axiom, the third and final step is to apply the second axiom to select the set of DPs which require the least amount of information content and have the greatest probability of success.

4.6.2.1 Feasibility: Satisfying the Design Constraints

As done previously for the top level DP, a table of criteria based on the design constraints may be generated to assess the feasibility of the lower level DPs . For example, the constraints listed in Table 4.7 are used determine the feasibility of the cooling method chosen to satisfy FR1: increase heat transfer coefficient. Given the thermal density and spatial restrictions, the only feasible solution is forced convection.

$DP1_d$: Forced convection with a heat sink.

Table 4.7. Constraints on DP1.

Constraint	Unit	Source
1. Thermal density of components	watts/cm^2	Component Team
2. Spatial limitations	cm	Solid Model
3. Maximum allowable air flow rate	CFM	Fan (DPx)

Table 4.8 below contains the set of criteria used to evaluate the DP chosen to satisfy FR2: reduce vibrations transmitted to the components. The packaging engineering assesses the feasibility of each DP based on the material cost, stresses and strains (using finite element codes) and manufacturing processes. Though this case study does not describe different manufacturing processes, it is important to point out that a mapping between the physical and process domains would be wise to fully assess the manufacturing costs of each proposed structure. Given the cost constraints and the manufacturing capability of the plant, the only feasible DP is the metal frame.

$DP2_a$: Sheet metal frame.

Table 4.8 Constraints on DP2.

Constraint	Unit	Source
1. Cost allocated to structure	$	Management
2. Maximum allowable stress applied to components	psi	Component Team
3. Weight of components	pounds	Component Team
4. Maximum weight of computer	pounds	Customer Needs

4.6.2.2 Design Matrix

A design matrix is generated to determine the level of functional coupling within the given set of DPs. Equation 4.1 below presents the resulting design matrix.

$$\begin{bmatrix} FR1 \\ FR2 \end{bmatrix} = \begin{Bmatrix} X & O \\ X & X \end{Bmatrix} \times \begin{bmatrix} DP1 \\ DP2 \end{bmatrix} \qquad (4.1)$$

FR1: Increase heat transfer coefficient

FR2: Reduce vibrations transmitted to the components

DP1: Forced convection with heat sink

DP2: Sheet metal frame

The geometry and weight of the heat sink (DP1) adds an inertial mass to the system and thus affects the vibration of the components. Therefore, the design matrix contains an X in the lower diagonal. When the design is decoupled, the design matrix dictates in which order the remaining FRs and DPs are to be developed. As a result of the functional analysis, the heat sink should be designed before the sheet metal frame.

If the design matrix were coupled, the design would not be acceptable. At that point, different DPs are investigated to satisfy the given set of FRs. The previous options of DPs were ruled out because they were simply not feasible - they did not meet the given constraints. If more than one DP were feasible, each set of feasible DPs is applied to the design matrix to determine the least functionally coupled set of DPs.

4.6.2.3 Information Axiom

Because the information Axiom is the last hurdle for the given DPs, it is used less frequently than the first axiom. However, if more than one set of DPs were viable after implementing the first axiom, the final selection is based on the second axiom. According to the second axiom, a product which requires less information content has a higher probability of success. Therefore, DPs which rely on many constraints are susceptible to failure because they require a higher information content.

4.7 Conclusion

Design constraints significantly affect the design process. A consistent and up-to-date set of constraints is important within a large and complex development effort in order to generate accurate FRs and assess the feasibility of DPs. The description (whether prescriptive or descriptive) and ramifications of DPs become system constraints at the lower level. Information about DPs is vital to populate the design matrix and consequently determines the level of functional coupling within the design.

A consistent representation system is required to capture design constraints. As the design becomes more complex, more constraints from different sources are introduced. Therefore, a method of updating all of the design constraints is crucial for making good decisions. A larger number of constraints also increase the complexity of the feasibility analysis. Questions such as which constraints should be considered or which set of constraints have a greater impact become more complicated and require special attention. A consistent representation of these constraints provides the framework to answer these questions and better organize the design process.

Chapter 5

Enhancing Representation within AD

This chapter investigates methods for improving the four types of knowledge representation (defined in Chapter 3) within AD. The structure of this chapter differs from Chapter 3 because each section of this chapter focuses on a specific domain of AD knowledge rather than a specific type of knowledge representation. The intent of this structure is to shift focus from types of knowledge representation to methods for enhancing knowledge representation within AD and the implementation of such proposed methods to an Axiomatic Design Software, presented in Chapter 7.

The discussion regarding each domain is divided into three knowledge types: relational, inferential and inheritable. Relational knowledge is used to describe the *links* between elements in different domains. Inferential knowledge is used to describe the *abstraction* of knowledge from elements within a domain and adjacent domains. Inheritable knowledge is used to describe the type of *structure* used to organize design elements within the domain.

5.1 Customer Domain

The customer domain, with respect to the other domains, has the least representation. Much more work is needed within this domain to improve the documentation of information flow from the customer domain to the remaining domains.

5.1.1 Inheritable Knowledge: A Proposed Hierarchy for the Customer Domain

Unlike other domains, the customer domain does not have a clear structure. The other domains possess a hierarchical structure in which new elements are entered as the design progresses. The growth of customer needs is not as intrinsic to the customer domain because the customer is not a real member of the design team.

Information in the customer domain may be structured in a hierarchy, in which the needs of the customer are sorted by categories defined by the designer[9]. A numbering system with a decimal system to represent decomposition, as proposed for the other domains, may be implemented. A natural prefix name for such elements is CN (Figure 5.1). The goal of the hierarchical structure is to identify high level needs of the customer and develop FRs to address each need.

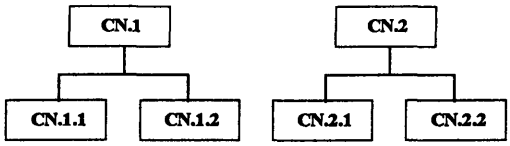


Figure 5-1. Customer domain hierarchy.

5.1.2 Relational Knowledge: Mapping from the Customer and Functional Domains

The mapping within the CN hierarchy and the functional domain does not need to be direct, as is the case between the remaining domains as dictated by the independence axiom. In addition, the zigzag process does not work well with the customer domain simply because it is not practical. Customers are interviewed at the outset and at subsequent intervals within the process to ensure the product meets all customer expectations. Unlike engineers, customers cannot be expected to be a part of the decisions made at every level within the design process.

Figure 5.2 below illustrates the mapping process between the customer and functional domains. The arrows in the figure indicate the FR which satisfied the CN in the customer domain. Because there is no restriction on the mapping process between the customer domain and other domains, CN elements may be referred to by any FR within the functional hierarchy.

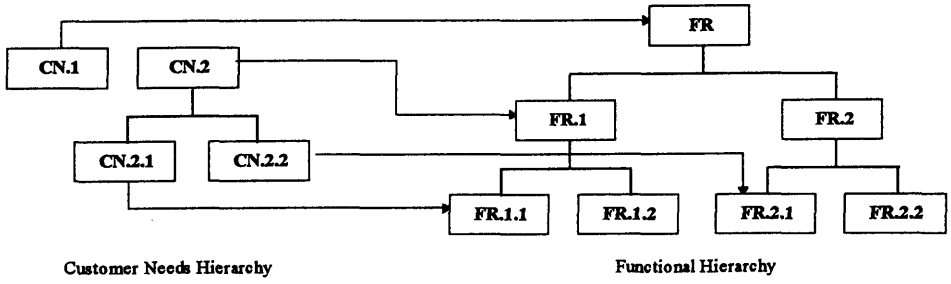


Figure 5-2. Mapping of CNs from the customer domain to FRs in the functional domain.

[9] Please refer to [29] for more examples and methods on structuring customer needs.

5.1.3 Inferential Knowledge: Obtaining Design Knowledge from the Customer

Inference is used to abstract information from customer needs and apply them to other domains and set of design constraints. Therefore, three different types of design knowledge may be inferred from the customer domain: 1) information required to map from the customer to the functional domain, 2) design constraints, and 3) customer suggestions for DPs and PVs.

5.1.3.1 Customer Suggestions

Customers may state their needs purely as customer needs or imply information in other domains, such as DPs and PVs. The additional information which they imply should be stored as part of a set of options within the appropriate domain. For example, suggested DPs should be stored as part of a DP database. During the FR-DP mapping process, the customer suggested DP may be recalled to satisfy the FR. The customer may even provide information about the process domain. For example, given the need to preserve our environment, the customer may be environmentally conscious and dictate the type of materials and chemicals used to manufacture the product.

5.1.3.2 Inference Knowledge to Map from the Customer to Functional Domains

The key to a successful design is to infer the right information from the voice of the customer. As might be suspected, this is a challenging task. In some cases, customers are aware of their needs and provide critical information regarding the functionality of the product or service. However, in other cases, customers cannot *verbalize* their need until they see or experience a product which satisfies their need. Needs which a customer cannot directly verbalize are called *latent needs* [29].

Latent needs are uncovered by reverse engineering the customer needs from customer accepted DPs. Examples of customer accepted DPs include DPs which the customer specifically provides or DPs that are currently successful in the market. For example, when a customer mentions a solution (instead of stating a need), it is the designer's task to abstract the need from this solution by first identifying the possible FRs of this solution and then the possible customer needs. Also, the designer may choose to uncover customer latent needs by benchmarking a customer accepted DP, such as a competitor's product. Thus, the designer may analyze the successful product by first identifying the FRs and then the CNs.

5.1.3.3 Constraints (Metrics) from Customer Needs

The last type of inferred knowledge is best illustrated in the case study presented in chapter 4. In section 4.1, constraints are abstracted from the given customer needs. In general, the process involves the use of metrics to quantify needs. Then, all resulting quantification which place a bounds on the design becomes design constraints. This process is illustrated Table 4.1 (customer needs) and Table 4.2 (customer based design constraints).

5.2 Functional Domain

The representation of knowledge within the functional domain is critical to problem definition. Therefore, the inheritance of information within the functional hierarchy, the relation of information to other domains, and the inference of information from functional elements are critical to the design process.

5.2.1 Inheritable Knowledge: Constraints and Functional Coupling

5.2.1.1 Inheritance of Constraint Information within the Functional Domain

The structure of functional requirements within the functional domain is critical to the development of new FRs and the traceability of information through the domain hierarchy. Chapter 3 presents the inheritance of constraint information within the functional domain, both directly within the functional hierarchy and as part of the zigzag process (Figure 3.2). Figure 5.3 below presents the top level FR and all of the input constraints from Figure 3.2.

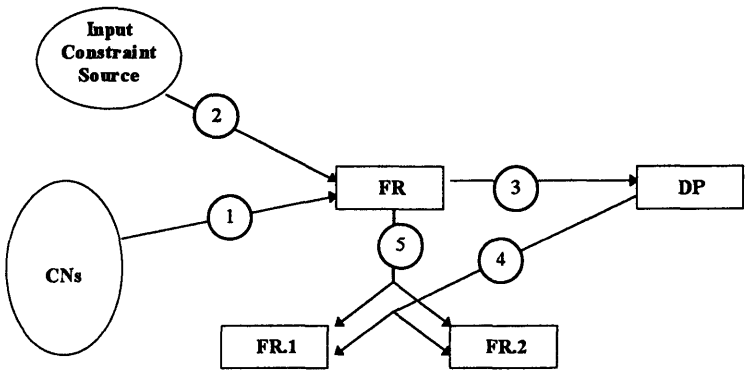


Figure 5-3. Constraint propagation within the functional hierarchy.

In Figure 5.3, constraint sets 1 and 2 are applied to FR. Constraints sets 3 and 5 are applied by the FR to its associated DP and sub FRs, respectively. Constraint set 5 represents the sum of all the input constraints, 1 and 2. However, constraint set 3 is a subset of the given constraints - the subset that is applicable for the DP selection.

It is clear that a numbering scheme is required for the set of design constraints in order to manage the information within the design process. It is recommended that each constraint is associated with its respective origin and have a number (C_1, C_2, ... , C_n). The format can be as follows:

C (source of constraint) number of constraint from source

For example, C (CN.3.1) 2 is the second constraint associated with element CN.3.1 from the customer domain. Constraints from the physical domain can be labeled accordingly, such as C (DP) 1 to represent the first constraint abstracted from element DP. Thus, if the source of the constraint changes or is removed, the affect of this change may be traced to the dependent constraints. Section 5.2.3 on inferential knowledge in the functional domain illustrates this representation scheme.

5.2.1.2 Inheritance of Functional Coupling from Decoupled Designs

Information Dependency of FR Development. Off-diagonal elements in decoupled design matrices indicate the information dependency of FRs on higher level DPs. For example, if an FR were coupled with several DPs, all sub-FRs require knowledge from all the DPs with which the parent FR is coupled[10]. Given the dependency of information in decoupled designs, a representation scheme is necessary to indicate which DPs should be considered when developing new sub-FRs.

If the decomposition of elements within the hierarchy were automated within a software environment, in which the user were able to enter FRs and DPs and decompose them, it would be necessary to advise the user which DPs should be considered when creating new sub-FRs. Because the design matrix stores this information, it would be redundant to create an additional representation scheme for this information. However, it is recommended that the numerical name of the DP(s) which is (are) creating the coupling should be passed down to all sub-FRs.

Design Sequence. FRs also inherit information on the design sequence, as to which FR should be decomposed first based on the nature of the design matrix. Therefore, each FR should be tagged to indicate the appropriate design sequence for these FRs. Thus far, there is no such representation scheme to order the FRs. One possible method is to place a number in parentheses after the FR decimal notation. For example, FR.1.2 (3) is the second FR in the vector but should be both satisfied by a DP and decomposed third relative to the remaining FRs in the vector. Renaming FR.1.2 as FR.1.3 is dangerous because it couples the identification of the FR with the order in which the FR should be satisfied. The identification and design sequence notations serve completely different purposes and should not be confused as one. If the design matrix were to change, and the order of the FRs were rearranged, renaming all of the FRs based on this result would lead to confusion.

The proposed sequence numbering scheme may be applied in design databases which store the design hierarchies. The programming and managing of such databases would become unnecessarily

[10] Please refer to Chapter 3 section 3.2.2.1 for more information on the affect of off-diagonal design matrix elements on the development of lower level FRs.

complex if the numeric names were changed for all elements within a tree. The proposed scheme would simplify the management of design data and any consequential changes made within the design matrices.

5.2.2 Relational Knowledge

Information within the functional domain is related to the customer domain, the physical domain, and the set of design constraints. A representation scheme is requires to associate the FRs with these domains of knowledge. A proposed method is to represent FRs as transforms, with inputs and outputs. These FR inputs and outputs may be partially defined by constraints from the customer domain, higher level DPs and PVs, and the set of design constraints.

The following are example system constraints: 1) a person is currently at point A, 2) the person needs to be at point B, and 3) within time C. This set of constraints can be used to formulate a new FR which states *transport a person from A to B* and within an associated FR constraint of *time C*. This FR may be modeled as a transformation in which the first constraint is an input to the transform and the second is an output of the transform. In addition, the set of constraints associated with the FR may be listed along with the FR (see figure 5.4 below).

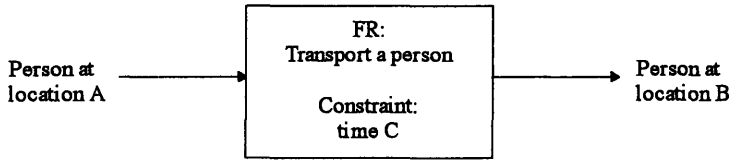


Figure 5-4. Representation of an FR as a transform.

The representation of an FR as a transform provides a concise method for describing FRs and helps derive the FR from given design information (i.e., constraints). In addition, the development of FRs is better documented if the origin of the constraints are documented. Thus, if there is a change in any of the constraints, the development of the FR can be reanalyzed. The representation scheme for constraints proposed in section 5.2.1.1 can help document the origin of constraints which are used to develop new FRs. Section 5.2.3.2 discusses this issue in further detail.

5.2.3 Inferential Knowledge

5.2.3.1 Generating Information from FRs to Select DPs

The representation of an FR as a transform, as shown in figure 5.4, simplifies the generation of information from FRs used to select viable DPs. The information necessary to obtain DPs is outlined by

the input and output information associated with the transform and the constraint(s) specified along with the FR.

Additional information may be obtained from the FRs from the source of the constraints which were used to develop the FRs. For example, as part of the example presented in section 5.2.2, the customer needs, higher level DPs, and/or higher level PVs may contain more information about the attributes of the person which is to be transformed. Attributes including the person's height and weight may be listed along with the source of the position constraint.

For example, the constraint which indicates the person's final position may be traced back to an original customer need. Assume the position constraint is defined as C (CN.2.1) 1: *position B*, where CN.2.1 is a person's need to attend an event. In addition, the CN may also have other associated constraints which are not a part of the FR definition, such as the person's attributes: C(CN.2.1) 2: *person's weight* and C(CN.2.1.) 3: *person's height*. Therefore, the source CN can provide additional information on the person being transported. This additional information may contain subtle points which were previously not considered as part of the FR and could thus help select a viable DP.

5.2.3.2 Developing FRs from Given DPs and Constraints

The case study in Chapter 4, section 4.5, illustrates how FRs may evolve from given constraints. For example, if a DP satisfies the intended FR, but does not meet some design constraint, a sub-FR as part of the DP system may remedy this problem.

A simple case. For illustrative purposes, we will consider a device which is intended to dry a wet surface. Thus the FR in this case is to *dry the surface* and the DP we will consider is *radiative heat transfer*. In addition, the FR may also have an associated temperature constraint, T. The DP may satisfy the FR, but it may not satisfy the given temperature constraint. If the resulting temperature value obtained after heating the surface, T_H, does not equal the surface temperature constraint, T, a new sub-FR is created to transform the surface temperature from T_H to T. Thus, if T_H is greater than the required T, the sub-FR is to cool the surface to the desired temperature. A sub-DP would be designed as part of the drying system to ensure the system satisfies the constraint. Thus, the drying system is decomposed and contains a sub-FR: *cool surface* as a consequence of the higher level DP: *radiative heat transfer*.

Proposed Representation Scheme. A representation scheme is required to indicate how each FR is created. For example, FRs created from constraints require a method of tagging the constraints to the new FR. If any of the constraints change, the FR could possibly change. Such a scheme is also required for FRs which are specified directly by CNs, DPs and PVs. A naming structure can help organize the development of FRs. Therefore, FRs can be listed with the information used to generate them. For example, FR.1 {CN.1.2} indicates that FR.1 originates from a customer need, CN.1.2. FRs can also originate from multiple sources. Thus FR.2.3 {DP.2 and C (CN.3.4) 3} indicates that FR.2.3 is a result of

DP.2 and the third constraint which originates from the customer. For example, the surface dryer presented above resulted in the development of a sub-FR as a result of a DP and a constraint and thus the sub FR could be tagged with the instigating DP and constraint. Thus, if any of the source elements change, the proposed representation scheme can help identify FRs which should be re-examined.

5.3 Physical Domain

5.3.1 Inheritable Knowledge: Constraint Propagation

As described in Chapter 3, section 3.2.2.2, inheritance of constraints is possible within all of the design hierarchies, including the DP hierarchy within the physical domain. As a review of this previous discussion, constraints may 1) propagate between domains through the zigzag process and 2) propagate through individual hierarchies independent of the zigzag process.

The following example illustrates the inheritance of constraints through the physical domain through the zigzag process between the functional and physical domains. In this example, the top level FR is to *write*. The FR has a set of associated constraints: 1) the writing is to be performed with a person's hand, and 2) the total weight supported by the hand should not exceed a specified weight, *w*. For the sake of this illustration, the constraints are assumed to originate from a customer need, CN, and thus the two constraints are labeled C(CN)1 and C(CN) 2, respectively. A *lead based system* is selected as the DP to achieve the FR (see figure 5.5). The hierarchy is decomposed and two lower level FRs are defined: 1) FR.1, *grip the device*, and 2) FR.2, *apply lead to paper*. A DP is selected for each lower level FR: 1) DP.1, *wood covering*, and 2) DP.2 *normal force applied by hand*.

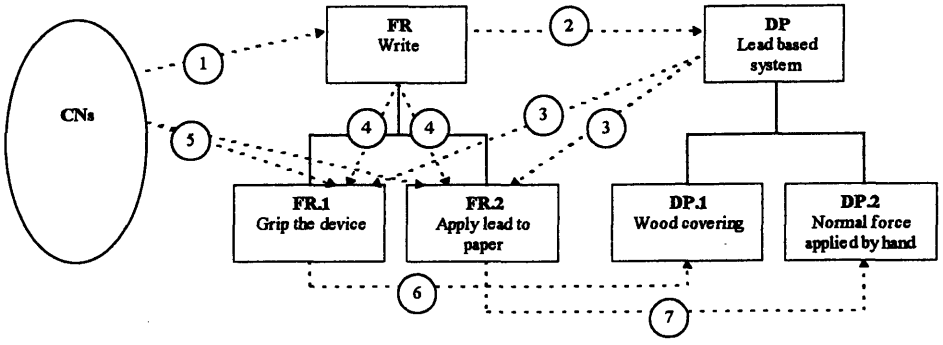


Figure 5-5. Decomposition of a pencil.

The dashed lines in Figure 5.5 indicate the possible transfer of constraints among the domains and elements. In this case, lines 1, 2 and 4 contain the same set of constraints - C(CN)1 and C(CN)2.

The top level DP may also input additional system constraints. The system constraints are applied to lower level FRs as shown by line 3 in the figure. For example, 1) the physical weight of lead and 2) the writing characteristics (the normal force required to achieve a certain quality of writing) are two system constraints from the top level DP which apply to lower level FRs. The representation of these new system constraints are C(DP)1 and C(DP)2, respectively.

Before developing new FRs, it becomes evident that more specific information is required from the customer domain to specify the *darkness* of writing required. This information may be obtained from the customer domain as a constraint which may be associated with the lower level FRs. This constraint is illustrated by line 5 and may be represented as C(CN)3.

Once each lower level FR is defined, the associated constraints for each FR are defined as well. Thus, for each FR, there is a set of system constraints which are imposed by higher level decisions. These constraints are illustrated as lines 3, 4, and 5 in the figure and originate from the higher level DP, higher level FR, and customer domain, respectively. The DPs associated with FR.1 and FR.2 must achieve the functionality under the given set of constraints. Lines 6 and 7 in the figure indicate the set of constraints each DP must satisfy.

DP.1 and DP.2 are assessed with respect to the set of design constraints. For DP.1: *wood covering*, the weight of the proposed DP cannot exceed the weight of the total device C(CN)1 minus the weight of the lead C(DP)1. Thus, the weight of DP.1 is determined by the set of constraints and is equal to C(CN)1 - C(DP)1. The resulting value of this analysis may be represented as a constraint or attribute of DP.1: C(DP.1)1. A constraint analysis is also required for DP.2:. *normal force applied by hand.* The given constraints applied to FR.2 are considered and a new constraint is obtained which indicates the minimum amount of force required to transfer the lead onto the paper. This minimum applied force is equal to the total force needed to write {the minimum force is a function of C(DP)2 and C(CN)2} minus the normal force applied by the weight of the device, C(CN)1. Therefore, the constraint associated with DP.2: C(DP.2)1 is equal to f{C(DP)2 and C(CN)2} - C(CN)1. In conclusion, 1) an analysis is carried out to determine if the DPs satisfy the given set of constraints and 2) the values of the specific constraints resulting from the analysis are represented as attributes of the DPs which may be used as system constraints on lower-level FRs.

Such a representation scheme has possibilities within a design database. The database would contain the source of each constraint, the analysis carried out to determine the feasibility of the DPs with respect to the design constraints, and the final attributes of the DPs. If there is a change in the design constraints, say due to modifications of the constraint source, the database would prompt the designer which DPs should be reconsidered.

5.3.2 Relational Knowledge

Information from the functional domain and the set of design constraints is used to select the appropriate set of DPs within the physical domain. In section 5.2.2, FRs are represented as transformations. The same approach may be used to model DPs. Therefore, a DP may be modeled as an object which achieves the transform or function defined by the FR. The transform or function may be achieved by a causal relation and a physical parameter intrinsic to the DP. *A DP may be modeled as a physical entity which achieves a certain set of transforms through intrinsic causal relations.* Such a method of modeling DPs may assist in searching for a DP for a given FR and to determine functional coupling based on the intrinsic causal relations.

For example, we shall examine a bar of steel as a DP. We may identify 3 functions which this DP may contain. These are 1) transfer heat, 2) support a load, and 3) indicate position. The bar may transmit heat, q, through one dimensional conduction given a temperature differential as governed by Fourier's Law,

$$q = -\frac{AK\Delta T}{L} \tag{5.1}$$

where A is the cross sectional area of the bar, K is the thermal conduction coefficient of steel, ΔT is the temperature differential along the bar, and L is the length of the bar in the direction of heat transfer. The bar could support a maximum load F in tension, as governed by the yield stress,

$$F = YA \tag{5.2}$$

where Y is the yield stress of the material and A is the cross sectional area. The bar, say when used as part of a precision machine, can also indicate position through its length L. The length is a function of thermal and stress constraints,

$$L = L_o + \frac{FL_o}{EA} + \alpha\Delta T \tag{5.3}$$

where L_o is the initial length of the bar, F is the force applied, E is the material's Young's modulus, A is the cross sectional area, α is the thermal expansion coefficient, and ΔT is the temperature differential. The DP may be modeled as a collection of these causal relations. Figure 5.5 below present this information in the form of a DP model.

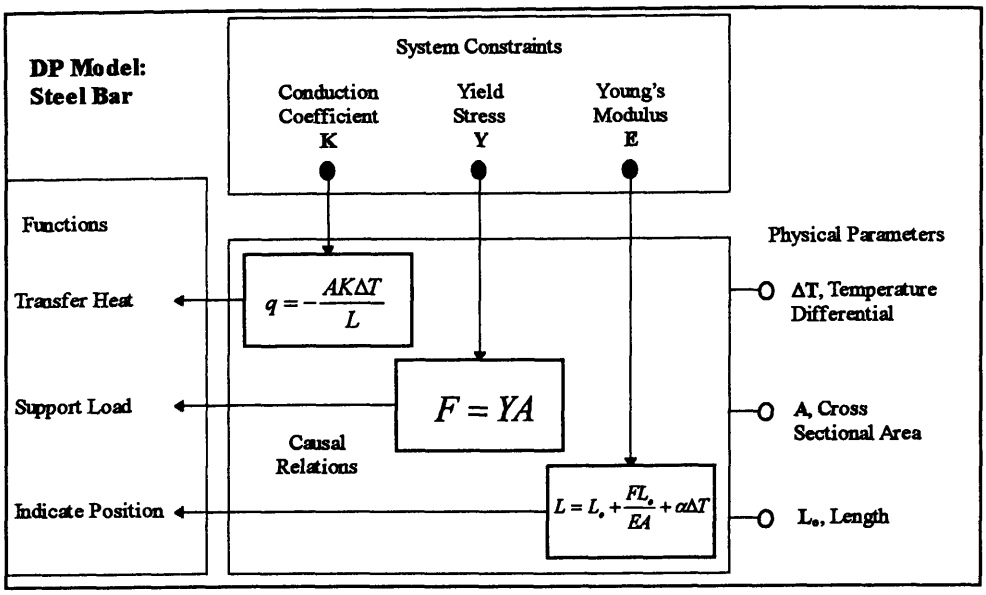


Figure 5-6. Transform Model of Steel Bar.

The model provides the set of functions which the physical object can achieve and the causal relations intrinsic to the physical object which dictate how the functions are achieved. The functions may be matched with FRs within the functional hierarchy such that a causal relation may be identified to achieve the FR. System constraints inherent to the modeled object, such as the material characteristics, are listed on the top portion of the model. Lines extend from each system constraint to a causal relation which requires the constraint value. The physical variables on the right are the potential DPs which can achieve the transforms on the left. Thus, each physical parameter is a DP which controls the value of a function through a causal relation. In conclusion, the proposed model can help search for DPs for a given FR and determine the relations within the design matrix based on the causal relations. The following example illustrates the development of the design matrix using the proposed model and causal relations.

5.3.3 Inferential Knowledge

Once a set of DPs is chosen to satisfy a set of FRs, the designer generates a design matrix to check for functional coupling. Information is obtained from the DP model and intrinsic causal relations to determine if there are any elements in the off-diagonal of the design matrix. Each element in the FR to DP design matrix A is defined as:

$$A_{i,J} = \frac{\partial FR_i}{\partial DP_i} \tag{5.4}$$

To continue with the previous example, we will state the following mapping of FRs and DPs:

FR_1	Transfer Heat	**DP_1**	Temperature gradient across bar
FR_2	Support Load	**DP_2**	Cross sectional area of bar
FR_3	Indicate Position	**DP_3**	Length of bar

The design matrix is generated using equation 5.4 and the causal relations shown in Figure 5.6.

$$\begin{bmatrix} FR_1: \text{transfer heat} \\ FR_2: \text{support load} \\ FR_3: \text{indicate position} \end{bmatrix} = \begin{Bmatrix} -\frac{AK}{L} & -\frac{K\Delta T}{L} & \frac{AK\Delta T}{L^2} \\ O & y & O \\ \alpha & -\frac{FL_o}{EA^2} & 1+\frac{F}{EA} \end{Bmatrix} \times \begin{bmatrix} DP_1: \Delta T \\ DP_2: \text{A} \\ DP_3: \text{L} \end{bmatrix} \tag{5.5}$$

In summary, the causal relations in the DP model provide information to generate elements in the design matrix. From this information and the resulting coupling, the designer may choose independent steel bars for each FR to remove the off-diagonal terms. Additional inferential knowledge is gained from the DP, including new system constraints. For example, the chosen DP values, such as the area, length, and temperature differential, become system constraints for lower level decompositions.

5.4 Conclusion

Representation methods are proposed for the customer, functional, and physical domains with respect to the relational, inferential, and inheritable knowledge contained within each domain. The representation methods are introduced to capture information which is a part of the axiomatic design process. The goal is to automate the process and the representation system through an axiomatic design software. The details of the software are discussed in Chapter 7.

Chapter 6

Case Study: Wafer Spin Coating

The following case study is based on a three week assignment in which the author visited a semiconductor process equipment company. The goal of the visit was to implement the AD method at the start of a new project and to therefore set the axiomatic design foundations for the project early in the conceptual phase. Work was underway on the next generation machine. Engineers worked on a scaled-up version of the previous model and required conceptual design analysis.

At the outset of the project, the author was given design specifications generated by the marketing department. The goal was to read through these documents and develop a high level decomposition of the machine. As part of this analysis, many constraints were gathered from the marketing document and applied to the various FRs within the high level decomposition. Due to the limited time of the work tour, the functional analysis was quickly focused on one sub-system of the design: the photoresist coating system.

6.1 The Lithography Process

The machine coats and develops wafers as part of the semiconductor manufacturing process. Silicon wafers, which are thin and round silicon disks, are used to make semiconductors, such as the processor chips in computers. Silicon wafers are cut from solid cylinders of silicon into thin sheets of about 0.5 mm thick. Then each sheet goes through a film depositon and oxidation process to properly dope the material to attain the desired electrical conductance. After which, the wafer goes through the lithography process, a process by which a pattern is transferred from a mask to the silicon surface (also known as substrate) [10]. The lithography process is illustrated in Figure 6.1 below.

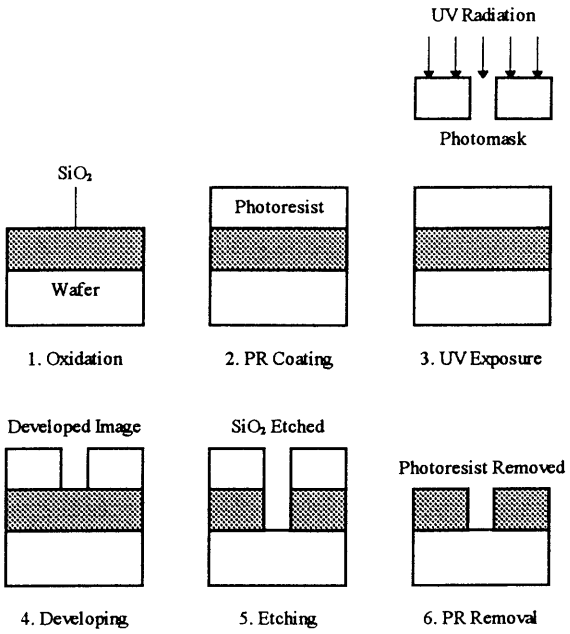


Figure 6-1. Lithography process (adapted from [10]).

At the outset of the lithography process, the wafer has a uniform oxide coating on its surface. The next step in the process is the application of a photoresist (PR) film which is sensitive to ultraviolet light. The application is achieved by applying a layer of 0.5 to 2.5 μm PR onto the wafer surface and then spinning the wafer at several thousand rpm for 30 to 60 seconds to provide uniform coverage. This is called the coating process. The next step is pre-baking in which the wafer is heated to remove the solvent in the PR layer and to harden it before UV exposure.

After the PR layer is hardened, a mask is placed over the wafer and then subjected to ultraviolet radiation. This process is known as exposure. The wafer is then developed to remove the UV exposed PR and a pattern remains on the wafer. A post baking process toughens the remaining PR pattern. The exposed oxide layer is then etched away and the PR is removed by dipping the wafer in a solvent solution. The final product is a wafer which contains the mask pattern in the oxide layer [10].

6.2 Design Objective

The focus of the case study is the coating of photoresist on the wafer surface - step 2 in the lithography process outlined in Figure 6.1. The company and other competitors produce coating systems but they are customized for 200 mm diameter wafers. The process technology currently allows 300 mm diameter wafers and thus industry in moving towards the larger size wafers to increase the number of semiconductor products produced per wafer and thus the overall throughput. The design objective is to

develop a scaled-up 300 mm version of the older 200 mm coating system. As part of the conceptual development phase, the coating system components are analyzed such that FRs are established and a design matrix is populated. The analysis is presented in the following sections and commences with the given design constraints[11].

6.3 Design Constraints

At the highest level, the design constraints can be classified into three categories: 1) customer, 2) corporate, and 3) coating process constraints.

6.3.1 Customer Constraints

Constraints are abstracted from the customer needs as defined in the marketing documents. The constraints range from the foot print of the machines (amount of floor space the machine occupies) to throughput of wafers processed per machine. The most critical customer constraint is the cost of the machine. Because their are several manufacturers of semiconductor process equipment machines, the new machine must be competitively priced. All such customer needs and associated constraints are part of the customer domain. The details of this domain are not listed to maintain confidentiality. Each constraint can be listed with the associated customer needs obtained from the marketing document in the form presented in Chapter 5, (C (CN.x) y), where x is the number of the CN and y is the number of the constraint associated with the specific CN.

6.3.2 Corporate Constraints

The corporate constraints include strategic and development issues. The company must develop the new machine in order to stay competitive and maintain its current loyal customer base. Thus, the company must sell a certain number of new 300 series machines starting at a given product release date. Business constraints also imply the development constraints, including the number of engineers employed to work on the project, the tools available for the engineers, and the time given to complete the project.

6.3.3 Process Constraints

The wafer processing constraints are provided by the process engineers and contain a list of constraints, ranging from temperature limits of the wafer surface to the necessary uniformity of the PR coating. For example, the wafer must be within a certain temperature range before and after each step in the lithography process.

---

[11] Please refer to [30] for more detailed information on the design of the 200 series system.

6.4 The development of top level FRs and DPs

FR: ***Coat wafer with photoresist.*** The top level FRs and DPs are developed from the information within the customer domain and the associated design constraints. For the scope of the lithography process chosen, the top level FR is to coat the wafer with photoresist. The associated constraints of this FR are the throughput, air properties, and the development time and development resources available. Specific constraints associated with the air properties include the maximum number of particles per cubic volume, temperature, humidity, and air dynamics (laminar non-recirculating conditions) near the wafer. As a note, laminar non-recirculating flow prevents the splashback of dry particles which may contaminate the coated wafer surface.

DP: ***Spin Coating.*** Spin coating is the current industry standard DP for uniformly coating a wafer with photoresist. However, research is being conducted on applying other coating techniques to PR coating. The aim is to reduce the amount of excess photoresist and increase the throughput of processed wafers. Examples of alternative coating techniques common to other industries include meniscus, capillary, patch, dip, and spray coating [20]. However, at this time, the technology for applying alternate coating techniques to wafer coating is not yet mature, and therefore, given the product release date, spin coating is the most feasible DP.

Spin coating provides additional system constraints to the design. These constraints include the rotation speed of the wafer, the ambient conditions surrounding the wafer, and the excess amount of PR which is spun-off the surface. These new system constraints are used to develop new FRs at the next level of decomposition.

6.5 The development of Second level FRs and DPs

Axiomatic design is more established for new designs, rather than scale-up designs which is the current application within this case study. However, the following analysis illustrates the method can prove to be useful in structuring given information and pin-pointing weak facets of a design.

6.5.1 Define FRs

The next level of FRs are highly dependent on the process constraints and the decision of using spin coating as the DP. The following section presents each lower level FR.

FR.1: ***Decrease Wetting Angle of PR.*** The wetting angle of the PR should be decreased to lower the wetting angle to improve the spreading and uniformity of the coat onto the wafer surface.

FR.2: ***Deposit Photoresist.*** A mechanism is needed to transfer PR from the PR reservoir to the wafer surface. The associated FR constraints are the rate and amount of PR required and the wafer location onto which the PR is deposited.

FR.3: ***Control Thickness.*** After PR deposition, the wafer spins at several thousand RPM to control the thickness of coating. An associated FR constraint is the total spinning time.

FR.4: ***Remove Solvent from PR.*** According to the lithography process, solvent is removed from the PR coat to harden the coat. Associated constraints are the ambient conditions surrounding the wafer during the removal process.

6.5.2 Search for DPs

After establishing the set of FRs, customer needs, and set of deign constraints, a feasible DP is sought for each FR. Each numbered DP in the following list satisfies the associated numbered FR in the previous section.

DP.1: ***Hot Plate Vapor Prime Chamber.*** A vapor prime is applied to the wafer surface to improve the wetting angle of the PR. The chamber contains a conductive hot plate which heats the wafer to the specified deposition temperature and a system for injecting the required vapor.

DP.2: ***Deposition Arm.*** An arm which contains a supply line of PR is positioned over the wafer such that the PR falls onto the wafer surface.

DP.3: ***Spindle.*** A motor driven chuck supports the wafer and rotates the wafer at the required RPM.

DP.4: ***Post Bake Hot Plate.*** A conductive hot plate, similar to the one used as a part of DP1, is used to heat the wafer to remove the solvent from the PR layer.

6.5.3 Functional Analysis

A functional analysis of the system results in the following design matrix:

$$\begin{bmatrix} \text{FR.1: Apply Vapor Prime} \\ \text{FR.2: Control Thickness} \\ \text{FR.3: Deposit Photoresist} \\ \text{FR.4: Remove Solvent} \end{bmatrix} = \begin{Bmatrix} X & O & O & O \\ O & X & O & O \\ O & O & X & O \\ O & O & O & X \end{Bmatrix} \times \begin{bmatrix} \text{DP.1: Hot Plate Vapor Chamber} \\ \text{DP.2: Spindle} \\ \text{DP.3: Deposition Arm} \\ \text{DP.4: Post Bake Hot Plate} \end{bmatrix}$$

The design matrix is uncoupled because the selected DPs do not provide any functional coupling. Therefore, this is an acceptable design. According to the spin coating design, each FR is satisfied independently of the other DPs.

6.5.4 Lower-level FRs and DPs

The two hot plates, DP.1 and DP.4, can potentially increase the total time required for the processing of a wafer because after each heating process, the wafer temperature must be lowered to the temperature required for the next process. Cooling the wafer by natural convection is not feasible because

it would not satisfy the given throughput constraint. Therefore, the wafer must be cooled after each heating process. This results in new lower-level FRs for each hot plate, DP.1 and DP.4. The new FR elements are to cool the wafer to the temperature required by the next process and within a time constraint that satisfies the overall throughput rate.

In addition, spinning the wafer propels excess photoresist off the wafer surface. Therefore, a method is required to collect the excess PR so that it may be reused. FR.3.1 is to collect excess photoresist. The proposed method of collecting the excess PR is to place a catch-cup around the wafer perimeters so that the PR splashes against the inner wall of the cup and drains down to the bottom where it is collected. Therefore, DP.3.1 is a catch-cup. The catch cup contains an efficient drainage system that returns the PR to its reservoir prior to drying. The details of the drainage system are sub-FRs of the catch-cup and are not discussed in this case study. The final coating system is illustrated in Figure 6.2 below.

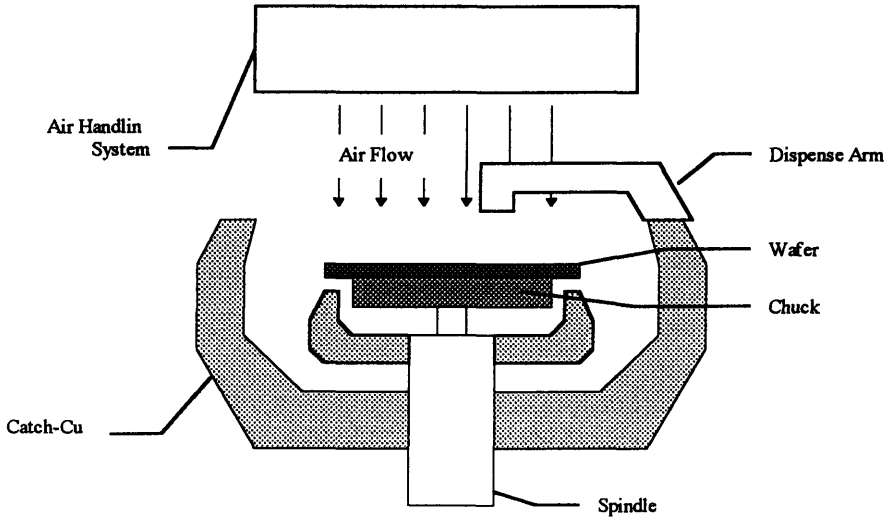


Figure 6-2. PR coating system.

Note, baking modules DP.1 and DP.4 are not a part of the coating system. The chuck supports the wafer and rotates it at several thousand RPM. A dispense arm moves along the wafer and dispenses the PR over the wafer surface. As the wafer rotates, excess PR is caught by a catch-cup which surrounds the periphery of the rotating wafer. An air handling system provides particle free air at the appropriate temperature, humidity, and laminar flow conditions.

The DP which achieves the given FRs under the time constraint is a conductive chill plate. therefore, each hot plate has a dedicated chill plate which the wafer is transferred to after baking. Figures 6.3 and 6.4 illustrate the functional and physical hierarchies, respectively, for the coating process.

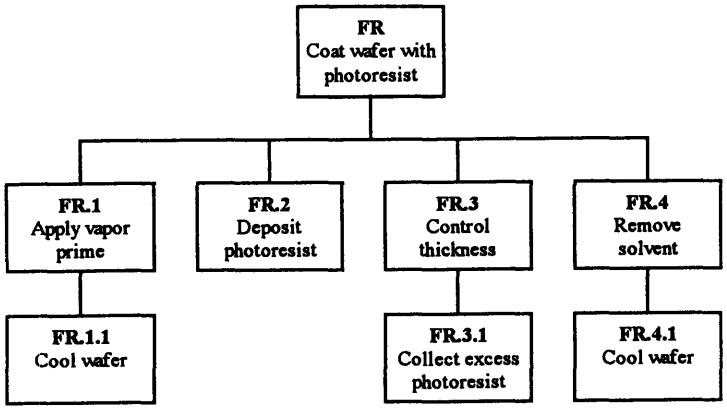


Figure 6-3. Function hierarchy of wafer coating process.

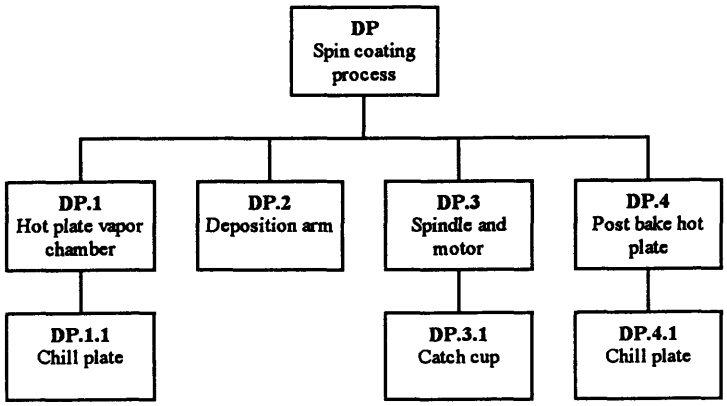


Figure 6-4. Physical hierarchy of wafer coating system.

6.6 Conclusion

The design analysis of the coating system illustrates the use of the axiomatic design process within a real development process. The case study documents the constraints, functional requirements, and design parameters. Such a design information structure is useful to understand all of the given constraints, develop a proper problem formulation, and brainstorm different options for each given requirement. In addition, the design matrix sets the framework to analyze the interactions between the individual components in the system.

Chapter 7

Development of an Axiomatic Design Software

Companies are turning to the computer and all its peripherals to communicate and document ideas. They are incorporating engineering data managers (EDM's) and intranet tools (an internet environment used within a company) to streamline the development process. These means of information transfer and graphical interfaces provide modern methods of communication and data storage within a design process. Thus, they are useful for storing and handling design drawings, vendor information, and project management issues. However, companies are focusing more on the hardware and software requirements to improve the design process, and less on the design process automated within this medium. Thus, much of these tools are lacking a sound design method. A design environment without a fundamental design process lacks 1) design tools, such as functional analysis, 2) a consistent design environment with one set of terminology shared by all the departments within the company, and 3) a real time creative design environment in which designers may create and document new designs as they gather existing data from sources available within the company.

This chapter presents the functional requirements and design parameters for a proposed software tool based on the axiomatic design process. The development of this software tool is presented within the AD framework. The following sections 1) define the design domains for the axiomatic design software, 2) provide preliminary customer needs, 3) identify high the FRs and DPs for the software, and 4) identify design constraints and their sources.

7.1 Software Development Design Domains

Before developing the software, the four design domains are identified. The customer, functional, physical, and process domains are outlined in the following sections and the source of information for each of these domains is specified.

7.1.1 Customer Domain

The customers of this software are designers working in small to large scale design projects. Therefore, the software is designed to handle single designers or a multiple number of designers working on large scale design projects involving several design teams. The specific source of customer information includes a number of design engineers currently working in industry, both in the US and Europe. The customers also include designers which are currently mechanical engineering students at the Massachusetts Institute of Technology. Information from these customers is vital for the successful development of the design software. Specific customer needs of such designers are detailed in section 7.2.

7.1.2 Functional Domain

The functional domain is specified by the engineers developing the software. Because the Axiomatic Design Software (ADS) development effort is part of a research project, the functional domain is currently specified by a team of graduate students and an advising professor, Professor Nam P. Suh, as part of the Axiomatic Design Group at MIT. Of the Axiomatic Design Group, two graduate student are working on the software development project, including Derrick Tate, a Ph.D. candidate, and the author of this thesis, Vigain Harutunian.

If the software development effort were to increase in scope, the development may occur within a software company so that the functional domain is specified by several teams of software engineers. The form of FR specification may include specific text describing system details, flow charts specifying information flow, and drawings depicting graphical interfaces. In general, the form of the FR specification should clarify the mapping between the FR domain and the physical domain (which contains the actual software).

7.1.3 Physical Domain

The physical domain consists of the software itself. Physically, the software is composed of scripts of code, sets of graphical interfaces, a data structure, a database, and other elements which help generate the software environment. This physical software requires information from all other domains. For example, the physical domain requires information from the customer domain to determine the user friendliness of the graphical interface and the type of computer platform the customer prefers.

The physical domain requires information on the function of the physical elements. This is the direct mapping from the elements within the functional domain. In addition, the physical domain obtains information from higher level decisions made within the process domain. As described in the following section, the process domain consists of processes required to fabricate or code the software. Information about the graphical and network abilities available, and time required to code and debug the software help define physical elements which compose the software.

7.1.4 Process Domain

The process domain consists of the resources required to code the software. Elements within the process domain include programmers (developers and debuggers), programming languages, graphical user interfaces, developer's tool kits, and debugging programs or methods. As is the case with other domains, the process domain requires information from other domains and the set of design constraints. In the case of the ADS development process, the team obtained process specific information from our immediate customer, our sponsor, as to which platform the software should run on. Our sponsor specified the UNIX environment. Given our hardware constraints, the team used the LINUX operating system which runs on PC compatible machines. Also, because of limited time constraints, the team decided to hire additional students as programmers and chose a developer's tool kit for the UNIX environment called Tcl/Tk to expedite the software graphical user interface (GUI) development process[12]. These process decisions were made with information provided by our customer and project constraints.

7.2 Customer Needs

The AD team has had the opportunity to work with several design engineers and students to develop and enhance the ADS. The following is a list of customer needs obtained from potential customers:

CN.1 Improve my ability to generate good designs
- CN.1.1 Improve the quality of my design
- CN.1.2 Help me find good solutions to suite my design needs
- CN.1.3 Help sort through design information which is currently in my head

CN.2 Reduce the number mundane tasks within design process
- CN.2.1 Easily access different solutions, such as those in catalogues
- CN.2.2 Improve the power or efficiency of current design tools

CN.3 Improve communication during design
- CN.3.1 Provide a consistent set of design data during the design process
- CN.3.2 Help breakdown departmental walls - enhance concurrent engineering
- CN.3.3 Easily access up-to-date information on current design
- CN.3.4 Improve project planning and task allocation

CN.4 Facilitate documentation of design information for future use
- CN.4.1 Provide a user friendly documentation interface
- CN.4.2 Minimize documentation time
- CN.4.3 Allow cutting and pasting between sections of old and new designs

[12] Please refer to [18] for more information on the Tcl/Tk developer's toolkit.

7.3 Top-Level FR and DP

From the above list of CNs, the software design team develops the FRs. To meet the CNs, the software team proposes the following top level FR:

FR: ***Provide an integrated design environment based on axiomatic design.***

The top level DP is any medium which satisfies this FR. For example, possible DPs include 1) DP_a: verbal communication of design knowledge through good management, 2) DP_b: a paper based documentation system containing a set of design guidelines (i.e.: ISO 9000), or 3) DP_c: a computer software environment with data retrieval and entry terminals for each engineer and a central database for storing design information.

The decision is DP_c, a software environment, because of the insufficiency of the other DPs. The other proposed DPs would not handle the large amount of information in today's complex systems. Also, the advancements in computer environments, such as computer power and user friendly interfaces, make DP_c a more fitting choice.

DP: ***An axiomatic design software environment.***

7.4 Constraints

System constraints imposed by the selected DP include the need for a computer terminal for each engineer, a networked commuter environment, and engineers that are willing to learn new computer tools. Input constraints are listed in Table 7.1 below.

Table 7.1. High-Level Input Constraints

Source	Number	Constraint
Customer	C (CN) 1	Communication protocols with computer analysis software used within design process (FEA, CFD, Solid Modeling, etc.)
	C (CN) 2	Computer platform favored by sponsor: UNIX
	C (CN) 3	Design process used within design teams
	C (CN) 4	Type of product being developed (Mechanical device, electrical circuit, software, etc.)
Development Team	C (DT) 1	Programming skills
	C (DT) 2	Number of developers (throughput of work)
	C (DT) 3	Research budget
	C (DT) 4	Knowledge of other design tools

7.5 Second Level FRs and DPs

The next level FRs are based on the information obtained from the higher level decisions, including the customer needs, and top-level FR and DP. The following are the next level FRs:

FR.1: ***Provide data storage (entry and retrieval).*** A means of entering and retrieving design information is critical for the design process. The constraints which apply to FR.1 include C (CN) 1, C (CN) 2, and all development team constraints.

FR.2: ***Guide designer through the design process.*** The software should guide the user through the axiomatic design process (steps outlined in Chapter 2). Constraints associated with FR.2 include C (CN) 3, C (CN) 4 and all development team constraints.

The following is the list of DPs selected to satisfy the above FRs:

DP.1: ***Perl programming language.*** The team chose Perl for database development because of the sponsor's platform needs C (CN) 2, the research budget C (DT) 3, and the need to access large amount of information quickly. Perl is a share-ware programming language, runs on the UNIX operating system, and provides data storage with fast data entry and retrieval [31].

DP.2: ***Tcl/Tk programming language.*** The team chose to use the Tcl/Tk programming language to develop procedures which guide the user through the design process. Tcl/Tk is share-ware programming language and developers toolkit which incorporates widgets for the development of a

graphical interfaces and execute in both UNIX and Microsoft Windows environments. It is high level and requires little programming experience. On the average, Tcl/Tk requires ten times less code than a pure C program with the same graphical functionality [18]. Thus, the language satisfies the customer and development team constraints.

Equation 7.1 presents the design matrix constructed for this level of FRs and DPs. The matrix is uncoupled because the DPs can be developed relatively independent of one another. At the highest level, the Perl database developer defines the protocol which is used to communicate between the graphical interface and the database. Once that protocol is defined, the two DPs are developed independently of one anther. The database programmer and graphical interface programmer require little interaction.

$$\begin{bmatrix} FR_1 \\ FR_2 \end{bmatrix} = \begin{bmatrix} X & O \\ O & X \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \end{bmatrix} \tag{7.1}$$

7.6 Third Level FRs and DPs

The following sections describe the decomposition of FR.1 and FR.2 and provide the DPs for each lower level FR. The next level FRs take into account the selection of the Perl and Tcl/Tk programming languages.

7.6.1 Decomposition of FR.1: Provide data storage (entry and retrieval).

Chapter 3 describes the information required within axiomatic design. Thus, the data storage system should accommodate inferential, relational, and inheritable knowledge. The fourth knowledge type, procedural knowledge, is achieved by FR.2. The FRs are:

FR.1.1: ***Input CNs, FRs, DPs, PVs, and associated design constraints (inferential knowledge).*** As new information is entered, each new element is justified by its source and any analysis or inference (if any) used to arrive at the new information. This software FR has a critical constraint, C(CN)1: the ability to access information from other analysis design tools.

FR.1.2: ***Define relational knowledge among elements in different domains.*** Elements within the same level of the design hierarchy require relations among elements in different domains. For example, relations are required within the design matrices representing the FR-DP and DP-PV mappings. In addition, relations are required between elements in the customer domain and the functional domain.

FR.1.3: ***Transfer inheritable knowledge within hierarchies.*** As described in chapter 5, knowledge, such as constraints, is inherited by lower level elements from higher level elements within the design hierarchies. The database should therefore allow the transfer of attributes from higher level elements to lower level elements.

The design team developed the following DPs to satisfy the above FRs.

DP.1.1: ***Variable structure.*** Each element is stored as a string of attributes. These attributes include the name of the element, numeric representation of the element (i.e., PV.3.5), the name of its parent elements, and its associated constraints.

DP.1.2: ***Relation structure.*** The relation between each element is defined as a new variable pointer which can be addressed as one of the attributes in the variable structure. Therefore, the variable structure affects the relation structure.

DP.1.3: ***Hierarchy structure.*** Lower level elements can obtain attributes, such as design constraints, from their parent elements. Therefore, as new elements are created under parent elements, the database transfers attributes from the parent variable to the child variable.

Equation 7.2 presents the design matrix which contains the functional coupling for this system. Element DP.1 affects the development of FR.2 and FR.3 because it naturally sets the framework for how elements are stored. Thus, if the variable structure changes, the relation and hierarchy structure must also change. Therefore, the variable is developed first, and the other two DPs are developed subsequently.

$$\begin{bmatrix} FR_{1.1} \\ FR_{1.2} \\ FR_{1.3} \end{bmatrix} = \begin{bmatrix} X & O & O \\ X & X & O \\ X & O & X \end{bmatrix} \begin{bmatrix} DP_{1.1} \\ DP_{1.2} \\ DP_{1.3} \end{bmatrix} \tag{7.2}$$

7.6.2 Decomposition of FR.2: Guide designer through the design process.

The axiomatic design process can become overwhelming for engineers that are not familiar with the AD process. However, the software can aid the user by walking through the process. The following are the critical functions which the software should provide:

FR.2.1: ***Guide user through the entry of CNs, FRs, DPs, PVs and design constraints.*** The software should guide the user through the zigzag process. The software should have a framework which prompts the designer for lower level elements and remind the designer what higher level decisions should be considered when generating lower level elements. As discussed in Chapter 3, the zigzag process requires information from the design matrices. Information on the relations among higher level elements can be accessed from the Perl database.

FR.2.2: ***Prompt user for propagation of constraints from previous levels.*** Constraints are inherited from previous levels as explained in Chapter 3 and 5. Constraints propagate from higher level CNs, FRs, DPs, and PVs. In addition, the specific higher level elements are determined by the hierarchy and the design matrices.

FR.2.3: ***Prompt user for selection of elements (relational knowledge).*** Before decomposing to another level, the software prompts user to check each chosen element with respect to the given set of design constraints, the first axiom, and the second axiom.

The DPs for the above FRs are generated with the Tcl/Tk developer toolkit. They are as follows:

DP.2.1: Graphics for the entry of *CNs, FRs, DPs, PVs, and design constraints.* A set of menus walk the user through the zigzag process and prompt the user to define elements in the domains. The entry of these items is carried out by an entry label and is shown pictorially as part of a hierarchy. The graphics indicate what design information should be considered when entering the new element (CN, FR, DP, PV or constraint). Figure 7.1 below illustrates the entry system of the ADS. Subsequent decompositions propose additional functionality to this interface.

DP.2.2: ***Graphics to show the propagation of constraints.*** A graphical box shows the list of constraints associated with each element and prompts the user to indicate which constraints should propagate down to new child elements within the hierarchy.

DP.2.3: ***Graphical automation of the first axiom.*** Once elements are entered for each mapping process, the software prompts the user to check the given elements with respect to the first axiom. Figure 7.2 below illustrates the design matrix within the current ADS.

Equation 7.3 presents the design matrix of this mapping. The design matrix is uncoupled because none of the DPs create a functional coupling outside the diagonal. Information required by each DP is obtained from the Perl database and is not taken directly from the graphical objects. Therefore, if any of the functions were to change, only the DP associated with the changing FR would be modified - the other DPs would not have to be altered.

$$\begin{bmatrix} FR_{2.1} \\ FR_{2.2} \\ FR_{2.3} \end{bmatrix} = \begin{bmatrix} X & O & O \\ O & X & O \\ O & O & X \end{bmatrix} \begin{bmatrix} DP_{2.1} \\ DP_{2.2} \\ DP_{2.3} \end{bmatrix} \tag{7.3}$$

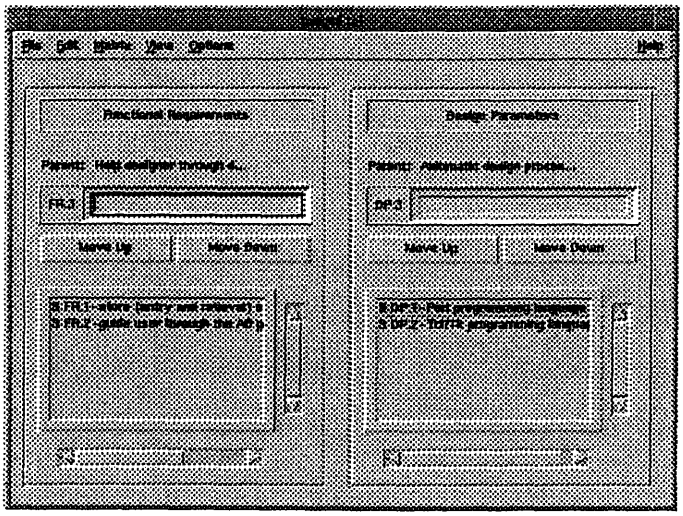


Figure 7-1. ADS entry window (DP.2.1) -Windows and UNIX compatible.

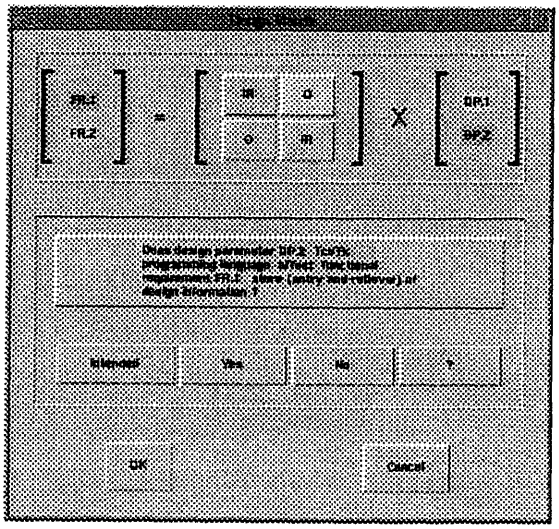


Figure 7-2. Design matrix window of ADS (DP.2.3).

7.7 Second Generation ADS

The following section presents the FRs for the second generation ADS which was in development during the writing of this thesis. The FR and DP decompositions continue from the previous section. The purpose of this work is to introduce the representation and constraint propagation issues presented in Chapters 3 and 5.

7.7.1 Decomposition of FR.2.1 and DP.2.1

The following are the sub-FRs for FR.2.1: Prompt user for entry of CNs, FRs, DPs, PVs and design constraints (inferential knowledge from previous levels).

FR.2.1.1: Walk user through the design process - specify what information is to be entered (CN, FR, DP, PV, or design constraint)

FR.2.1.2: Allow user to input and edit information in design domains

FR.2.1.3: Display information in the design hierarchies

FR.2.1.4: Display which higher level information should be considered when defining the current element

FR.2.1.5: Allow user to enter the specific information used to justify the current element

The following are the sub-DPs for DP.2.1: Graphics for the entry of CNs, FRs, DPs, PVs, and design constraints.

DP.2.1.1: A set of windows which tell the user what information should be entered

DP.2.1.2: A window with an entry label

DP.2.1.3: An interactive graphical hierarchy

DP.2.1.4: A list of higher level elements and their attributes which should be considered per given element

DP.2.1.5: An element justification window which can select attributes from higher level elements or design constraints as a justification for establishing the current element.

Equation 7.4 presents the design matrix for the given system of FRs and DPs. The matrix is uncoupled because each DP can be built separately. Once each DP is called, it relies on the Perl database for information and does not affect the functionality of the other DPs. Figures 7.3, 7.4, and 7.5 illustrate the proposed interface for the AD template.

$$\begin{bmatrix} FR_{2.1.1} \\ FR_{2.1.2} \\ FR_{2.1.3} \\ FR_{2.1.4} \\ FR_{2.1.5} \end{bmatrix} = \begin{Bmatrix} X & O & O & O & O \\ O & X & O & O & O \\ O & O & X & O & O \\ O & O & O & X & O \\ O & O & O & O & X \end{Bmatrix} \begin{bmatrix} DP_{2.1.1} \\ DP_{2.1.2} \\ DP_{2.1.3} \\ DP_{2.1.4} \\ DP_{2.1.5} \end{bmatrix} \qquad (7.4)$$

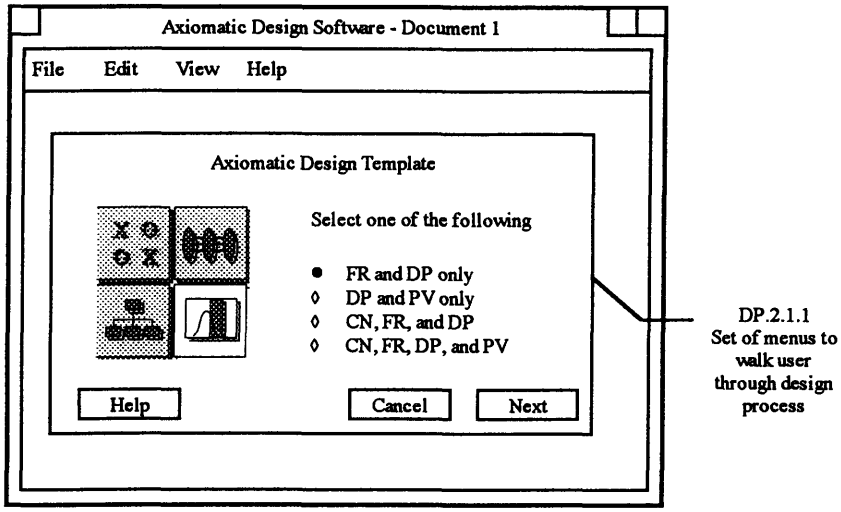


Figure 7-3. Opening window to Axiomatic Design Software with template running.

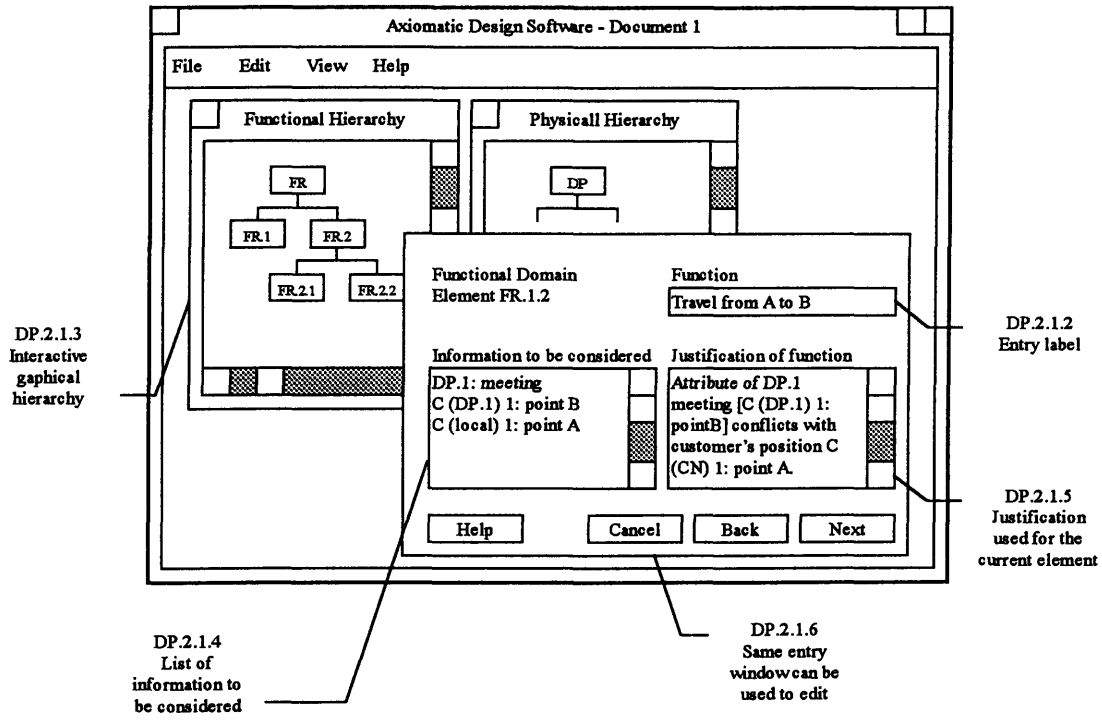


Figure 7-4. Window for FR entry.

The following are the set of DPs under DP.2.2: Graphics to show the propagation of constraints.

DP.2.2.1: A list of constraints which could potentially propagate to the current element

FR.2.2.2: A button which allows the user to select from the list of constraints

FR.2.2.3: An entry label

FR.2.2.4: A list of constraints selected

Equation 7.5 presents the design matrix for the above set of FRs and DPs. The matrix is decoupled because DP.2.2.1 (list of potential constraints) dictates how the constraints are selected, (FR.2.2.2). Figure 7.6 shows the proposed graphical interface for the constraint propagation function.

$$\begin{bmatrix} FR_{2.2.1} \\ FR_{2.2.2} \\ FR_{2.2.3} \\ FR_{2.2.4} \end{bmatrix} = \begin{bmatrix} X & O & O & O \\ X & X & O & O \\ O & O & X & O \\ O & O & O & X \end{bmatrix} \begin{bmatrix} DP_{2.2.1} \\ DP_{2.2.2} \\ DP_{2.2.3} \\ DP_{2.2.4} \end{bmatrix} \tag{7.5}$$

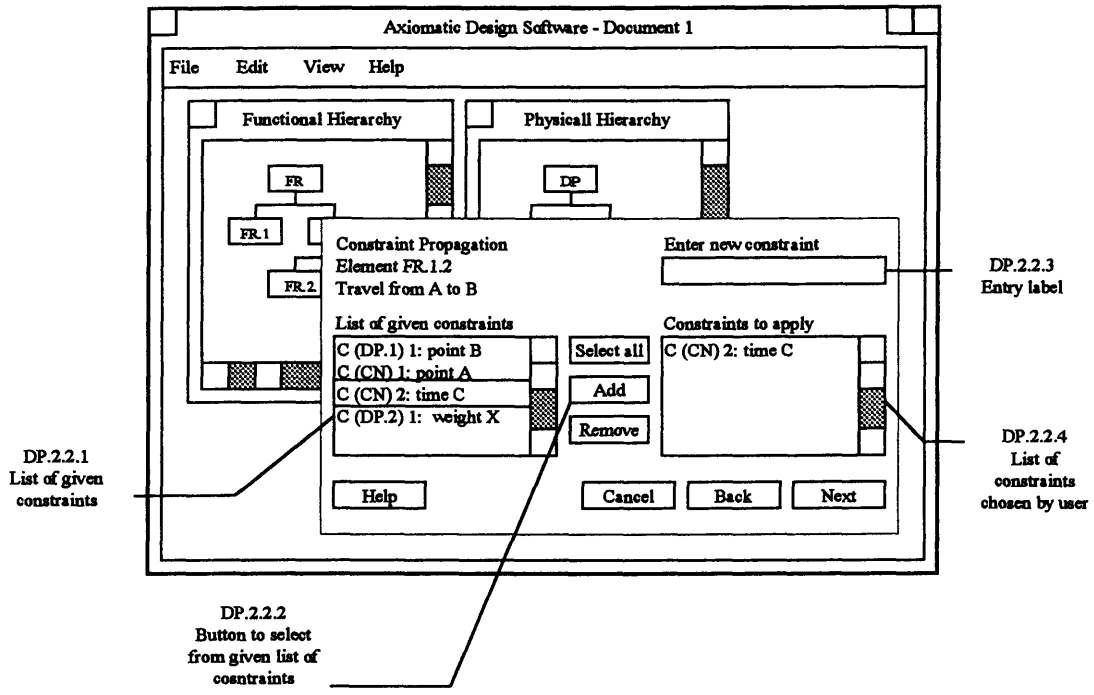


Figure 7-6. Constraint propagation window.

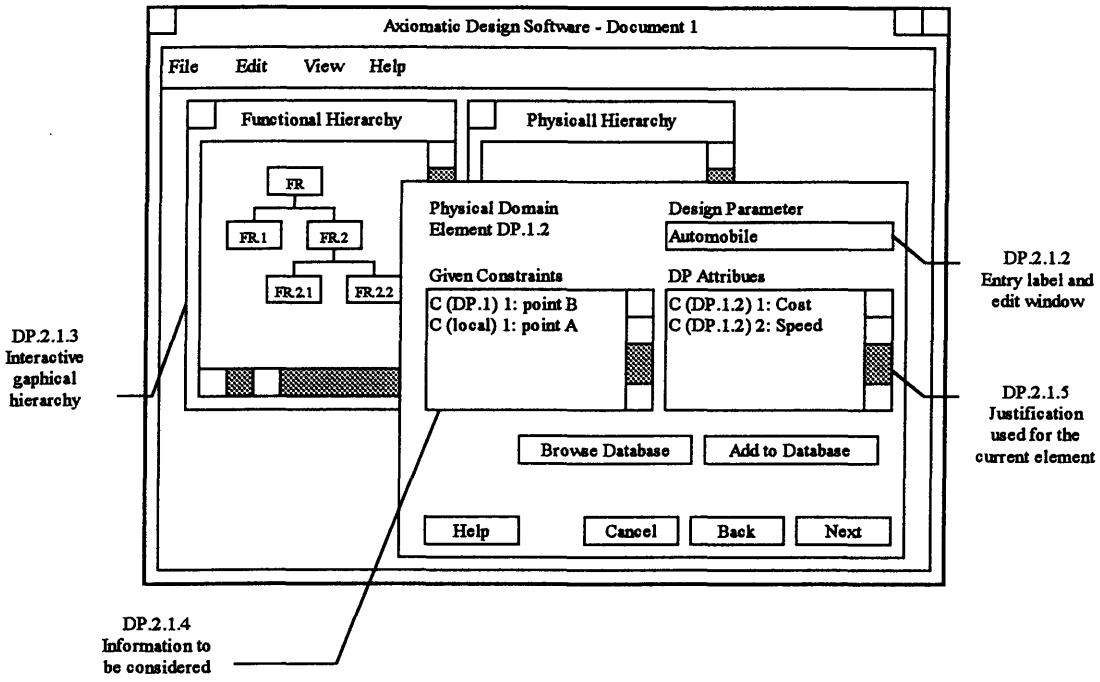


Figure 7-5. Window for DP entry.

The windows walk the user through the different phases within the AD process. The user has the option to turn the windows off and go through the process manually. The graphical hierarchy allows the user to double-click on any element and the appropriate window to enter new information or edit the current information. The DP database referred to in Figure 7.5 allows the user to search for possible DPs that satisfy the given FR. Tools for checking the given DPs with respect to the design constraints and axioms are described in the following section.

7.7.2 Decomposition of FR.2.2 and DP.2.2

The following are the set of FRs under FR.2.2: Prompt user for propagation of constraints from previous levels.

FR.2.2.1: Show the set of constraints that may propagate to the current element from each possible source

FR.2.2.2: Allow user to select the appropriate constraints which should propagate to the current element

FR.2.2.3: Allow user to enter new constraints

FR.2.2.4: Show user the constraints selected

7.7.3 Decomposition of FR.2.3 and DP.2.3

The following are the set of FRs under FR.2.3: Prompt user for Prompt user for selection of elements.

FR.2.3.1: Show the left and right hand vectors

FR.2.3.2: Allow user to enter an X and O relations in design matrix

FR.2.3.3: Inform user what each relation in the matrix represents

FR.2.3.4: Allow user to enter a justification for each relation

FR.2.3.5: Re-order the elements to obtain the least number of elements above the diagonal

The following are the set of DPs under DP.2.3: graphical automation of the first axiom.

DP.2.3.1: Graphical element containing vector numerical value and name

DP.2.3.2: Interactive buttons in the design matrix.

DP.2.3.3: A dialogue box to indicate what each relation represents

DP.2.3.4: A text box to enter comments on each relation entered

DP.2.3.5: Procedure to check all permutations of the matrix

Equation 7.6 presents the design matrix for the set of FRs and DPs. The matrix is almost uncoupled except for two elements. DP.2.3.2, the button which indicates which relation is being entered, provides information to satisfy FR.2.3.3 and FR.2.3.4. The information in DP.2.3.3 and DP.2.3.4 is therefore dependent on the information from DP.2.3.2. Figure 7.7 illustrates the design matrix interface.

$$\begin{bmatrix} FR_{2.3.1} \\ FR_{2.3.2} \\ FR_{2.3.3} \\ FR_{2.3.4} \\ FR_{2.3.5} \end{bmatrix} = \begin{Bmatrix} X & O & O & O & O \\ O & X & O & O & O \\ O & X & X & O & O \\ O & X & O & X & O \\ O & O & O & O & X \end{Bmatrix} \begin{bmatrix} DP_{2.3.1} \\ DP_{2.3.2} \\ DP_{2.3.3} \\ DP_{2.3.4} \\ DP_{2.3.5} \end{bmatrix} \quad (7.6)$$

The software executes the permutation procedure after the matrix is populated. Once the user presses *next*, the software calculates the sequence of DPs that results in the least number of elements above the diagonal. The sequence of the DPs resulting from the design matrix is stored in the database and is used by the software at the next level of decomposition to determine which FR is decomposed first.

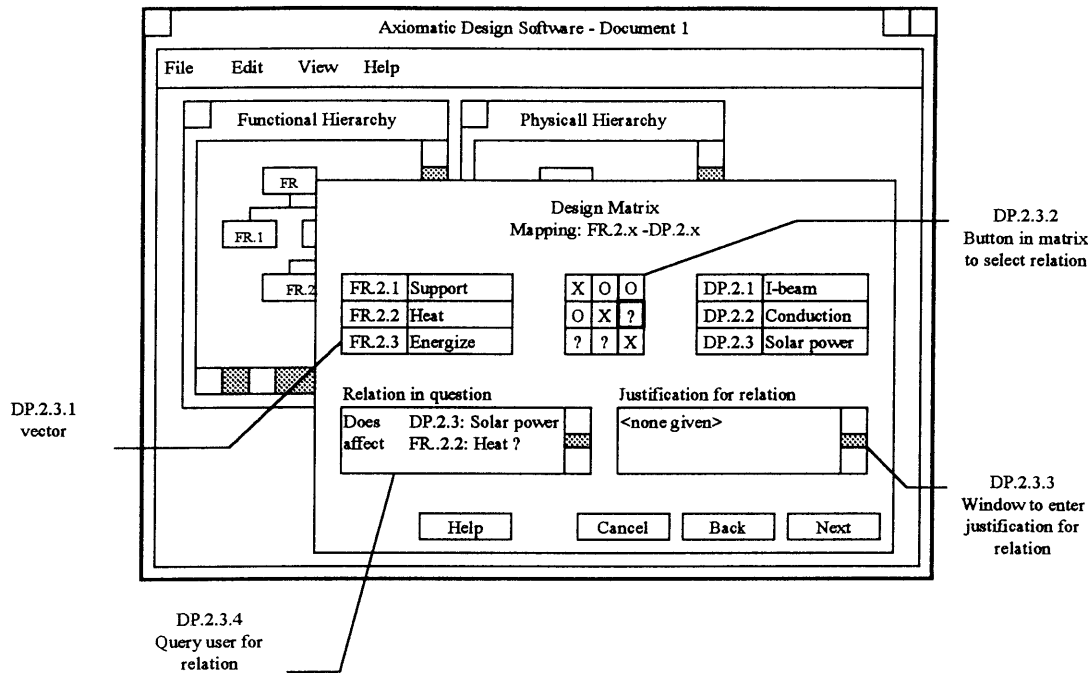


Figure 7-7. Design matrix window.

7.8 Conclusion

An axiomatic design software has been developed which runs in both UNIX and Microsoft Windows environments. The first generation software allows the user to enter information in the domains and the design matrix. The second generation system contains a set of windows to guide the user through the axiomatic design process and manages constraints along the hierarchy. An ultimate goal of the Axiomatic Design Research Group at MIT is to develop a Thinking Design Machine (TDM) which will automate the design process[13]. The representation scheme and the development effort of the ADS are a step towards the TDM.

As with any activity which requires the definition of functionality and the selection of objects to achieve the functionality, the AD method is successful for software development. The AD process is used effectively to identify the functions and associated DPs of the software. The use of the first axiom helps identify coupling among elements and thus dictates the sequence in which coupled DPs are developed.

[13] Please refer to [26] for more information on the thinking design machine.

Chapter 8

Conclusions

A representation scheme is sought to capture design knowledge within an axiomatic design software (ADS). Four types of knowledge representation are identified to establish a framework in which representation within AD is analyzed. Representation methods to enhance the documentation of information within the domains are developed and presented. The methods are implemented as part of the ADS. The effort indicates the need for a structured representation scheme based on a complete design process. A method which incorporates knowledge from the customer domain and transfers it to the other domains is vital for the management and documentation of a design project.

8.1 Enhanced Representation within AD

An enhanced representation scheme captures the rationale behind the information within the design domains and helps automate the design process. For example, lower level FRs are developed based on information from the higher level DPs and design constraints. A better representation of the higher level DPs and constraints improves the analysis and documentation of the lower level FRs. In addition, a more concise representation of DPs provides more information which may be used to populate the design matrix.

If there is a change in the design information, such as in design constraints or customer needs, the change propagates through the justification of each element as part of the proposed representation scheme. The software automates this process and prompts which lower level decisions are affected by the change. Thus, the representation shame improves the consistency of information within the domains throughout the design process.

8.2 Axiomatic Design Software

A software has been developed to help automate the axiomatic design process. At the current stage, the software relies on the user to enter information within the domains. The two main functions of

the software are to allow the user to store and retrieve design information within the framework of axiomatic design and to walk the user through the axiomatic design process. The software operates in both Windows and UNIX environments and consists of a graphical interface which guides the user through the process.

8.3 *Future Work*

The representation scheme presented in this thesis is only one step towards capturing information within the design process. In order to develop a complete Thinking Design Machine (TDM), there is also an additional need to develop methods to organize information outside the specific design project. For example, further research is required to incorporate methods which organize information on different types of customers, information on design constraints (such as industry standards) information on previous engineering design goals (FRs), information on possible DPs (such as a field specific set of DPs), and information on possible process variables (such as different manufacturing methods).

8.3.1 Proposed set of Domain Specific Databases

Within the context of the ADS, a general set of databases may be developed for each type of domain specific knowledge which can be connected to the design process, as shown graphically in Figure 8.1. The arrows in the figure indicate information flow to and from the design process because the set of databases can 1) provide information to the designer during each design project and 2) gather new information the designer provides to expand the databases.. Such a set of domain specific databases can possibly allow the designer to tap into a wealth of knowledge which may be applicable to the current design project.

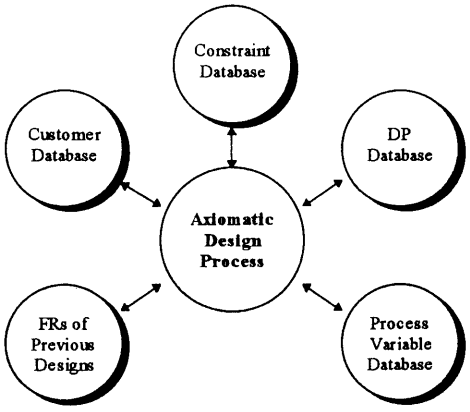


Figure 8-1. Domain Specific/Project Independent Databases for the design process.

8.3.2 Application of Representation Methods to ADS and Databases

However, the databases only contain domain specific information and are useless if they are not properly connected to the design process. The structure of information within the databases and the means of entering and searching for information within the databases are dependent on the design process representation scheme. Therefore, the relational, inheritable, and inferential representation methods presented in this thesis can help bridge the information between these databases and the design process. Thus, the next step within this research effort is to incorporate the representation methods proposed in this thesis to the ADS and develop a set of domain specific databases which can provide and gather information during the design process. Ultimately, the goal is to automate this process such that the computer automatically searches for information within the databases and consequently develop new FRs.

References

[1] Burke, E.; Elliman, D.; Heard, M.; "XCODAMS: An Engineering Design System Based on Constraint Propagation", *Artificial Engineering in Engineering, Computational Mechanics Publications*, 1994.

[2] Chang, C.; Lee, R.; *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

[3] Chomsky, N.; *Language and Thought*, Moyer Bell, Wakefield, Rhode Island, 1993. ISBN 1-55921-076-1

[4] Clausing, D.; *Total Quality Development*, ASME Press, New York, 1994. ISBN 0-7918-0035-0

[5] Eastman, C.; et.al.; "A Data Model and Database Supporting Integrity Management", *Computing in Civil Engineering*, Vol. 1, pp. 517-524, 1995.

[6] Evbuomwan, N.F.O.; Sivaloganathan, S.; Jebb A.; "A Survey of Design Philosophies, Models, Methods and Systems", (in press—*Proceedings of the IMCHE Journal, Part B: Design and Manufacture*, 1996)

[7] Glass, A.; Holtz, N.; Rasdorf, W.; "A System for Describing Design Artifacts Using the Knowledge Representation Techniques of Frames", *Engineering with Computers*, Vol. 10, pp. 197-211, 1994.

[8] Hakim, M.; Garrett, J.; "Modeling Engineering Design Information: An Object-Centered Approach", *Computing in Civil Engineering*, n. 1, pp. 563-570, 1994.

[9] Harutunian, V.; Nordlund, M.; Tate, D.; Suh, N. P.; "Decision Making and Software Tools for Product Development Based on Axiomatic Design Theory", *CIRP Annals*, Vol. 45/1, 1996.

[10] Kalpakjian, S.; *Manufacturing Engineering and Technology*, Addison Wesley Publishing Company, 1995. ISBN 0-201-53846-6

[11] Karnopp, D.; Margolis, D.; Rosenberg, R.; *System Dynamics, A Unified Approach*, John Wiley & Sons, Inc., New York, 1990. ISBN 0-471-62171-4

[12] Krishnamurthy, K.; Law, K.; "Towards a Formal Model of Version and Configuration Management for Collaborative Engineering", *Engineering Data Management: Integrating the Engineering Enterprise*, ASME, pp. 21-32, 1994.

[13] Kusiak, A.; "Dependency Analysis in Constraint Negotiation", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 9, pp. 1301-1313, 1995.

[14] Lindholm, D.; "New Application Areas of Axiomatic Design", 3rd CIRP Workshop on Design and the Implementation of Intelligent Manufacturing Systems, Tokyo, Japan, June 19-21, 1996.

[15] Molina, A.; et. al.; "A Review of Computer-Aided Simultaneous Engineering Systems", *Research in Engineering Design*, Vol. 7, pp. 38-63, 1995.

[16] Nordlund, M.; *Ph.D. Thesis* (forthcoming), Department of Manufacturing Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, 1996.

[17] Nordlund, M.; Tate, D.; Suh, N. P.; "Growth of Axiomatic Design through Industrial Practice", 3rd CIRP Workshop on Design and the Implementation of Intelligent Manufacturing Systems, Tokyo, Japan, pp. 77-84, June 19-21, 1996.

[18] Ousterhout, J.; *Tcl and the Tk Toolkit*, Addison-Wesley, Professional Computing Series, Reading, MA, 1994. ISBN 0-201-63337-X

[19] Pahl, G.; Beitz, W.; *Engineering Design*, (edited by K. Wallace), Springer-Verlag, New York, 1988. ISBN 0-387-50442-7

[20] Parodi, M.; et. al.; Spin Coating and Alternative Techniques for flat panel displays", Semiconductor International, Vol. 19, n. 1, 1996.

[21] Pecht, M.; *Handbook of Electronic Package Design*, Marcel Dekker, Inc., New York, 1991. ISBN 0-8247-7921-5

[22] Rich, E.; Knight, K.; *Artificial Intelligence* ,McGraw Hill, Inc., New York, 1991. ISBN 0-07-052263-4

[23] Rumbaugh, J.; *Object Oriented Modeling and Design*, Englewood Cliffs, New Jersey, 1991. ISBN 0-13-629841-9

[24] Suh, N. P.; "Axiomatic Design of Mechanical Systems", *Transactions of the ASME. Special 50th Anniversary Design Issue*, Vol. 117, pp. 2-10, June 1995.

[25] Suh, N. P.; *The Principles of Design*, Oxford University Press, New York, 1990. ISBN 0-19-504345-6

[26] Suh, N. P.; "Design of Thinking Design Machine", Annals of the CIRP, Vol. 39/1, pp. 145-148, 1990.

[27] Tate, D.; Nordlund, M.; "Synergies Between American and European Approaches to Design", *Proceedings of the First World Conference on Integrated Design and Process Technology (IDPT-Vol. 1)*, Society for Design and Process Science, Austin TX, pp. 103-111, December 7-9, 1995.

[28] Tate, D.; *Ph.D. Thesis* (forthcoming), Department of Mechanical Engineering, MIT, Cambridge, MA, 1997.

[29] Ulrich, K.; Eppinger, S.; *Product Design and Development*, McGraw-Hill, Inc., New York, 1995. ISBN 0-07-065811-0

[30] Van Doren, M.; "Precision Machine Design for the Semiconductor Equipment Manufacturing Industry", *Ph.D. Thesis*, Department of Mechanical Engineering, MIT, Cambridge, MA, 1995.

[31] Wall, L.; Schwartz, R.; *Programming Perl*, O'Reilly and Associates, Inc., 1991. ISBN 0-937175-64-1